ára: 560 Ft CD melléklettel

OS/2® Times

II. évf 4. szám A magyar OS/2 felhasználók lapja 97/4



Fejlesztés

Fejlesztőeszközök

Programnyelvek

Játékok

Ön is lehet programozó

VisualAge for BASIC

Kész program 90 másodperc alatt !

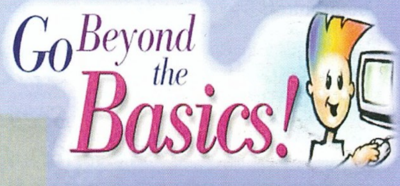
VisualAge for Java

CD

VisualAge for Java - Entry !

VisualAge for BASIC - 60 napos

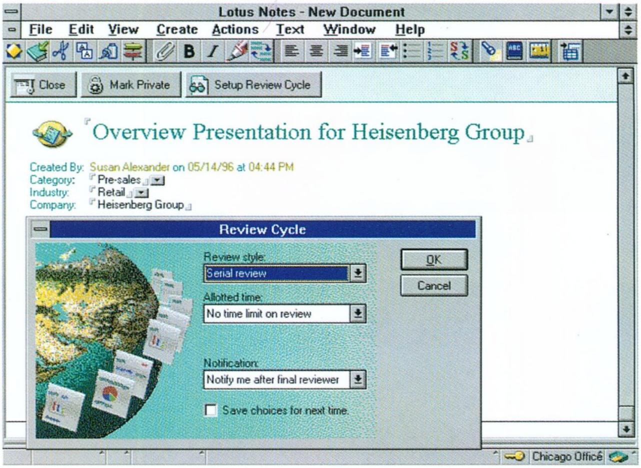
Photo>Graphics 1.04 - 30 napos



Független számítástechnikai folyóirat

Lotus Notes 4

világelső az irodaautomatizálásban



A Lotus Notes képes az iratok, dokumentumok, információk, ötletek – első pillanatban kaotikusnak tűnő – halmazában rendet teremteni, miáltal gyorsan, rendszerezetten férhetünk adatainkhoz, világszínvonalú biztonságtechnikával.

A Lotus Notesszal egyszerűsíthető az ügymenet, elkerülhető a többszörös munka, javítható a kommunikáció, s eddig nem tapasztalt hatékonysággal dolgozhatnak együtt a csoportok.

A Lotus Notes a világ első és messze legelterjedtebb csoportmunka szoftvere.

A Notes teremtette meg a csoportmunka fogalmát. A felhasználó cégek közül sokan minden üzleti információt a Notes segítségével tárolnak és továbbítanak. Ennek a sikeres szoftvernek a 4. verziója érkezett meg most, és csak az alkalmat várja, hogy Önnél is csatasorba állhasson.

A Lotus Notesszal kapott 25 mintaalkalmazáson és a Magyarországon hozzáférhető számos magyar alkalmazáson kívül bármikor kifejleszthető az Ön saját igényeit lefedő program, néhány héten belül.

A tipikus alkalmazások: **Iktatás, Ügyfélkezelés, Projektkövetés, Dokumentum nyilvántartás és keresés**

A Lotus Notes főbb tulajdonságai: **Fejlesztés, Beépített elektronikus levelezés, Replikációs technológia, Kifinomult biztonsági rendszer, Ügymenet (workflow) tervezés, Internet kapcsolat, Gyors bevezetés, Papír alapú dokumentumok kezelése.**

A Lotus Notes rugalmas: alkalmazkodik megszokott szövegszerkesztőjéhez, szoftvereihez, meglévő hálózati protokolljaihoz és operációs rendszereihez. Válassza Ön is a Lotus Notest. Keressen bennünket:

SZÜV TISZA Kft.

Szeged, Jobb fasor 6-10.

Tel.: (62) 432-332 Fax: (62) 432-445

E-mail: szuvszeg@tiszanet.hu

Szabó Illés SW manager

Egy éves az OS/2 Times

Amikor több mint egy évvel ezelőtt, 1996 áprilisában nekiláttunk terveket szövögetni egy újságról, amely végre kegyeibe fogadja az OS/2-t, még mi sem láttuk előre, mi lesz belőle. Hajtott minket a lelkesedés, hogy kedvenc operációs rendszerünk elfoglalhassa méltó helyét a hazai sajtóban, és bíztunk saját szakértelmünkben, hogy az újságot meg tudjuk tölteni megfelelő tartalommal. Első számunk 1996 szeptemberében, hosszú előkészítés után jelent meg, és számos gyerekbetegséggel küzdött. Azóta, köszönhetően a kétfordulós lektorálásnak (a szerkesztők és Kovács István olvasószerkesztő munkájának), Tóth Ferenc tördelésének, Nagy Tamás grafikai munkáinak, valamint számos újságtól érkező konstruktív kritikának a Rózsa Gábor által tervezett első újságra épülő belső elrendezés apró módosításokkal úgy alakult, hogy egyre jobban megfeleljen egy stabil lapról kialakított képnek. Az átalakításban mindig az olvasók által beküldött véleményekre támaszkodunk, így természetes, hogy egyre több dicsérő levelet kapunk. Szinte a kezdetek óta nagy mértékben segíti az olvasókkal történő kommunikációt Kádár Zsolt által gondozott internetes oldalunk (nélküle már rég káoszba fulladt volna honlapunk), ahol a levelezésen kívül a leggyorsabb és legbiztosabb módon rendelhető meg az OS/2 Times, illetve jóval a nyomtatott megjelenés előbb olvashatók újdonságaink, valamint az újság tartalma (és néhány cikk).

A helyes döntések sorát mi sem jelzi jobban, mint az, hogy az év eleje óta eladott példányszámunk másfélszeresére, az első újságárusi terjesztésben is megjelenő szám, az 1996/3-as decemberi kiadás óta pedig duplájára nőtt.

1997 áprilisában a növekvő eladásokra támaszkodva újabb fába vágtuk a fejszénket, megjelentettük első CD-ROM mellékletünket. Ezzel olvasóink is új és hatalmas lehetőséghez jutottak. Az időkorlátos Warp 4 melléklet után folyamatosan következnek értékükben egyre jobb és nagyobb korongjaink. A júniusi CD melléklet számos elismerést gyűjtött be, és bátran mondhatom, Magyarországon az egyik legjobb felülettel rendelkezik. Az OS/2 és a HTML nyelv lehetőségeit kihasználva, reméljük, egyre kényelmesebbé tesszük a böngészést, reményeink szerint nem csak OS/2 felhasználóknak, hanem más platformokon is. Nem lehet tehát véletlen, hogy az áprilisi szám óta eltelt időszakban megduplázódott előfizetőink száma!

Az első születésnap nemcsak időbeli változást hoz. Borítónkkal is jelezzük azt a váltást, amit a lapszerkesztés és kiadás terén végrehajtunk. Elsőként megszabadulunk a cikkírás nemes, de annál terhesebb hagyományától, az ingyenességtől. Cikkíróink eddig úgy mint mi, lelkesedésből dolgoztak, mostantól — reményeink szerint — folyamatosan emelkedő díjazással honoráljuk szakértelmüket. A lap belső szerkezetében is átalakul, a hangsúlyosabb területeket kiemeljük, illetve egy-egy témáról bővebben írunk. Ahhoz, hogy mindezt olvasóink számára változatlan áron biztosítsuk, kihasználjuk a digitális média kínálta lehetőséget, és számos cikket, anyagot a CD-n teszünk olvashatóvá. Jelenlegi számunkban is a nyomtatott újság terjedelmét meghaladó méretben találnak rajta cikkeket, és a jövőben is egyre több információt nyújtunk majd elektronikus formában, már nem egyszerű mellékletként, hanem az újság „meghosszabbításaként". Lapunk a megnövekedett anyagmennyiség ellenére továbbra is kéthavonta jelenik meg, így tudjuk biztosítani ugyanis, hogy olvasóink a megfelelő színvonalat kapják minden egyes számtól és CD válogatástól.

Jó olvasást és böngészést kívánok mindenkinek az új OS/2 Times hasábjain és a CD könyvtáraiban!

Ambrózy Gábor

Shareware regisztráció forintért!

Az OpenBlue Bt. augusztus végén indítja útjára shareware regisztrációs szolgáltatását. Ennek keretében a legnépszerűbb shareware programokat forintért regisztrálhatják a hazai felhasználók. A programok közül először az InJoy tárcsázóprogramot kínáljuk, egyből az eredeti ár feléért, *csak* magánszemélyeknek:

***2300 forint + ÁFÁ-ért!**

A program regisztrált felhasználói magyar elektronikus dokumentációt kapnak szeptember folyamán! Legyen Ön is jogtiszta program tulajdonosa, regisztráltassa használt shareware programját forintért!

Ha további adatok érdeklik, vagy más shareware programját regisztráltatná, forduljon munkatársunkhoz, Kovács István Attilához a 292-5988-as telefonszámon, szeptember 1-je után!

(* A program árát az Ft/USD árfolyam befolyásolhatja.)

Előfizetői akció!

Ajándéksorsolás előfizetőink számára!
Az IBM Magyarország ajándékaként összesen

2 darab OS/2 Warp 4
10 darab OS/2 könyv
5 darab OS/2 póló

kerül kisorsolásra.

A sorsolásban történő részvétel feltétele, hogy augusztus 19-én érvényes előfizetéssel rendelkezzen, vagy október 26-ig előfizessen az OS/2 Times magazinra.

Aki augusztus 19-én érvényes előfizetéssel rendelkezik, az október 14-én megjelenő számunkból már értesülhet is róla: megnyerte-e az egy darab Warp 4, a négy darab OS/2 könyv és a két darab OS/2 póló egyikét. (A sorsolást október elsején tartjuk és az eredményt az Interneten is közzétesszük!)

Ha nem nyerte meg az első fordulóban, akkor még mindig megnyerheti a következőben!

Aki ugyanis október 26-án érvényes előfizetéssel rendelkezik, az a december elsején megtartandó sorsoláson egy OS/2 Warp 4, hat darab OS/2 könyv, vagy három darab OS/2 poló egyikének nyertese lehet!

Ne feledje! Velünk az előfizetési díj többszörösét takarítja meg!

Szerkesztőségünk levelezési címén korlátozott számban igényelhető
VisualAge for Java 1.0 for Windows béta
szoftver.

OS/2® Times
A magyar OS/2 felhasználók lapja

Megjelenik kéthavonta az OpenBlue Bt. kiadásában.

Főszerkesztő, lapszervezés, hirdetés:
Ambrózy Gábor (amby@eik.bme.hu)
tel: 221-0019

Internet, hírek:
Kádár Zsolt (kadzsol@xs4all.nl)

Előfizetések, shareware:
Tóth Ferenc (etus@alarmix.net)
tel: 292-0157

Olvasószerkesztés, terjesztés:
Kovács István (kofa@math.bme.hu)
tel: 292-5988

Szerzők:
Ambrózy Noémi
Gervai Péter (grin@hajdu.hungary.net)
Horváth Zénó
Molnárka György (mgyuri@alarmix.net)
Morvai Zoltán
Nagy Attila (billly@caesar.elte.hu)
Nagy Tamás Gábor (bigtom@eik.bme.hu)
Rick Papo (rpapo@msen.com)
Sándor Gábor (gsandor@mol.hu)
Varga László (vargal@eik.bme.hu)

A szerkesztőség és kiadó levelezési címe:
1443 Budapest Pf. 256.

Felelős kiadó: Ambrózy Gábor

Tördelés: Tóth Ferenc

Grafika: Nagy Tamás Gábor

Nyomda: Zalai Nyomda Rt., Zalaegerszeg

A lap ára: 560 Ft CD melléklettel

Előfizethető a kiadó levelezési címén és a következő bankszámlaszámon:
OTP 11714006-20370972

Éves előfizetési díj:
2800 Ft CD melléklettel

Terjesztés: hírlapárusoknál és számítástechnikai szaküzletekben

HU ISSN 1416-4566



Először is szeretnénk megköszönni az olvasóknak a CD-vel kapcsolatos sok visszajelzést. Ezeket próbáltuk minél nagyobb mértékben figyelembe venni a mostani mellékletünk elkészítésekor, reméljük sikerrel.

A felhasználói felület kisebb módosításon esett át, Nagy Tamás, ha lehet, még szebbé tette. De nem csak a felületen változtattunk, hanem nyitottunk a Windows platformot is használók felé. A Windows 95/NT alatt a behelyezett CD automatikusan elindítja a Netscapet (már telepítve kell lennie) és megjeleníti a főlapot, ráadásul a CD \WINDOWS alkönyvtárába egy Windows alatt működő INF fájl megjelenítőt helyeztünk, reméljük, hogy hasznát veszik.

Természetesen erre a CD-re is nagyon sokminden került (csak a főbb programokat emelem ki):

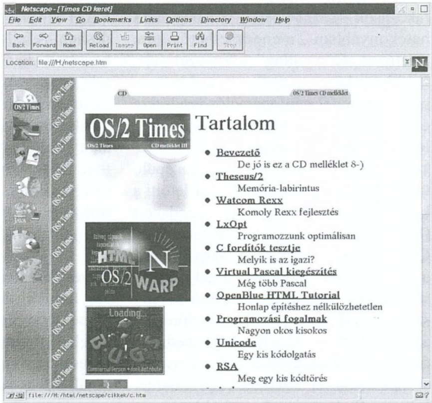
- o **VisualAge for Basic** 60 napos változat
- o **VisualAge for Java** Entry (sikerült betartani az ígéretünket :)
- o **StarOffice 4.0** béta 1 (augusztus 31-én lejár, de sajnos még nem volt újabb)
- o **Photo>Graphics** 1.04 30 napos változat
- o **Sibyl** demó
- o **Virtual Pascal** demó
- o **Java 1.1.1**
- o **BIOS Survival Guide**
- o **OS/2 API Project**
- o **VisualAge C++ 8-as FixKit** (ez annyira friss, hogy a CD-n a rá vonatkozó hivatkozásokat a szó szoros értelmében nem volt időnk átírni, így ott még 7-esként emlegetjük)
- o **Post Road Mailer** *ingyenes* verzió
- o **GhostScript**
- o **XFree86 3.3** és hozzá X alatt futó alkalmazások
- o Teljes **EMX+GCC**
- o **Magyar 29-es FixPak** javított verzióban

A hónap témájához kapcsolódóan a CD mellékletre főleg fejlesztőeszközök, fejlesztéssel kapcsolatos segédprogramok és leírások kerültek. Természetesen emellett más témájú programokat is raktunk a CD-re: játékokat, újabb Musthave programokat, eszközvezérlőket és egyéb segédprogramok stb. Reméljük mindenki könnyedén megtalálja a számára fontosakat.

Lapunk kis terjedelme miatt nem fért el minden cikk a nyomtatott újságban. Ezért úgy döntöttünk, hogy meghosszabbítjuk a lapot a CD-vel. Nincs ördöngösség a dologban, az eddig meglevő HTML felületbe tördeltük a helyhiány miatt a lapból kimaradó cikkeket. Annál is inkább tettük ezt, mivel a nyomtatott lapba szinte csak fejlesztéssel kapcsolatos anyag gyűlt és nem akartuk, hogy a téma iránt nem annyira érdeklődőknek ne legyen olvasnivaló. A CD-re került cikkeket feltűntettük az újság tartalomjegyzékében is a könnyebb kereshetőség érdekében.

A fentebb kiemeltek mellett természetesen egyéb hasznos dolgok is felkerültek a CD-re, kellemes mazsolázást!

Tóth Ferenc

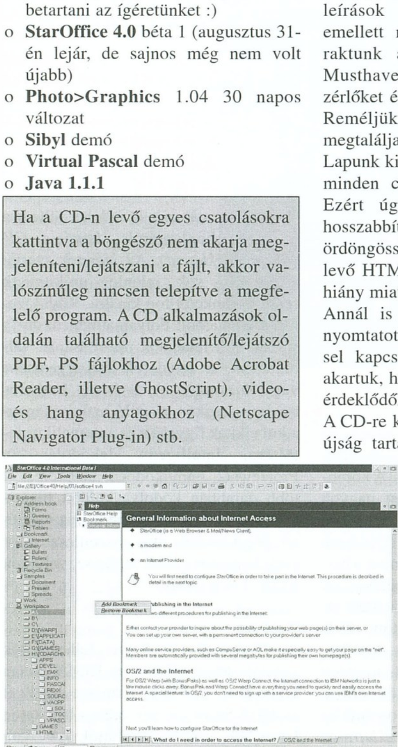


Ha a CD-n levő egyes csatolásokra kattintva a böngésző nem akarja megjeleníteni/lejátszani a fájlt, akkor valószínűleg nincsen telepítve a megfelelő program. A CD alkalmazások oldalán található megjelenítő/lejátszó PDF, PS fájlokhoz (Adobe Acrobat Reader, illetve GhostScript), video- és hang anyagokhoz (Netscape Navigator Plug-in) stb.

Az install.cmd első futtatásakor az OS2.INI fájlba beírja a CD meghajtó betűjelét. Ehhez a programoknak a CD-ről történő futtatásához/telepítéséhez van szükség. Amennyiben később másik CD olvasóból szeretné a lemezt nézegetni, akkor futtassa ismét az install.cmd-t!

Az install.cmd még két parancsfájlt másol be a merevlemezre. Amennyiben megtalálja az előző CD két fájlját (TIMESRUN.CMD, TIMESVIW.CMD), akkor azokat frissíti, ha nem, akkor kéri egy olyan alkönyvtár megadását, amely a PATH-ban található. Ha esetleg valamilyen ok miatt ez meghiúsul és semmit sem tud a CD INF fájljaiból futtatni/telepíteni, akkor kérjük másolja be Ön az X:\RUNTIME\TIMESRUN.CMD és X:\RUNTIME\TIMESRUN.CMD (X: a CD meghajtó) fájlokat egy, a PATH-ban levő alkönyvtárba!

Az install.cmd végül ellenőrzi két DLL meglétét: SO32DLL.DLL és TCP32DLL.DLL. Ezeknek a fájloknak telepítve kell lenniük a böngészők futtatásához. Mindkét fájl megtalálható az OS/2 TCP/IP részében, illetve az BonusPak csomag Internet részében. Ha még nincsenek telepítve, akkor az install.cmd bemásolja őket a LIBPATH-ba.



Sajnos az előző CD mellékletre két termék is hibásan került fel: a StarOffice 3.1, illetve a magyar 29-es FixPak. Minden CD mellékletünket alaposan teszteljük, ennek ellenére előfordulhatnak hibák. A javításokat mindig megtalálja honlapunkon, vagy következő CD mellékletünkön.

Bluebird: Az OS/2 család új tagja? ***

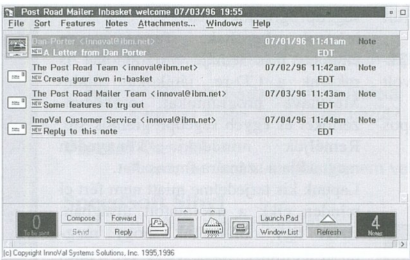
Az IBM várhatóan 1998 elején jelenik meg a piacon Bluebird kódnevű nevű termékével, amellyel elmozdul a ma hagyományosnak tekinthető ügyfél/kiszolgáló (client/server) modelltől a várakozások szerint sokkal olcsóbban és hatékonyabban menedzselhető, ún. "Network Computing" környezet felé. Ennek a környezetnek az a lényege, hogy az ügyfél gép operációs rendszere és a futtatandó alkalmazások a kiszolgálón, vagyis egy központi gépen helyezkednek el. Az Intel processzorra épülő ügyfél gép az indításkor először letölti a kiszolgálóról az operációs rendszert (Bluebird esetében ez természetesen az OS/2, a kiszolgáló pedig a Warp Server), a felhasználói felületet (amely igény szerint egy alkalmazás, pl. Web-böngésző vagy akár a komplett és előre beállított WPS is lehet), majd pedig a futtatandó alkalmazásokat. Amennyiben az operációs rendszernek vagy valamelyik alkalmazásnak újabb verzióját kívánják telepíteni, avagy konfigurációs változtatásokat kívánnak végrehajtani, akkor azt csak egyszer, a kiszolgálón kell megtenni. A Bluebird az operációs rendszerbe integrált JAVA-motor segítségével futtatni fogja az OS/2, DOS és Windows 3.x programok mellett a JAVA programokat is. A Citrix WinFrame-et a kiszolgálóra telepítve a futtatható alkalmazások köre kiterjeszthető a Windows 95 és Windows NT alá írt programokra is. Ezzel a Bluebird lényegesen több alkalmazást lesz képes futtatni, mint a Microsoft által tervbe vett Windows Terminal, amely gyakorlatilag csak az NT programokat kezelné. A híresztelések szerint az IBM a Bluebirdben az OS/2 fájlrendszerét, valamint a rendszer magját, a kernelt is tökéletesíti, s az operációs rendszerbe titkosítási technológiát is integrálnak. A béta változat talán még ez év végén megjelenik.

Visual Age for Java 1.0 **

Az IBM kiadta a Visual Age for Java első verzióját. Az Entry verzió ingyenes az OS/2 felhasználók számára és a http://www.software.ibm.com/ad/vajava oldalról tölthető le. *(Lásd a CD mellékleten!)* A továbbfejlesztett (Professional) változatot 84.95 USD-ért kínálja a Java SuperStore (http://www.javasuperstore.com), s létezik nagyvállalatok számára készített kiadás (Enterprise Edition) is. A vizuális fejlesztőeszköz hardverigénye meglehetősen magas, mivel Pentium processzorra és 32-64MB RAM-ra van szükség a megfelelő működéshez. A fejlesztőeszközzel egyidőben bocsátotta ki a Taligent cég (http://www.taligent.com) az IBM megbízásából készített VisualAge WebRunnert, amelynek segítségével igen rövid idő alatt lehet Web-alkalmazásokat fejleszteni. Tovább könnyítheti a Java programozók munkáját az Oberon Software (http://www.OberonSoftware.com) által piacra dobott Java Toolkit is.

Ingyen Post Road Mailer **

Az InnoVal Systems Solutions ingyenessé tette Post Road névre hallgató levelezőprogramját. Az ingyenes kiadás teljesen egyenértékű az eddig pénzért kínált 2.5-ös verzióval. A jótékonykodás oka az, hogy a gyártó cég rövidesen kiadja a program újabb, JAVA nyelven megírt változatát. További információk: http://www.innoval.com/free. *(Lásd a CD mellékleten!)*



1. ábra: Post Road Mailer: A kapott levelek

Java 1.1 for OS/2 Warp **

Megjelent a JDK (Java Development Kit) OS/2-es változatának 1.1-es bétaverziója (Service Level 1.1.1). A korábbi, 1.02-es változathoz képest nagyon sok újítást vezettek be, ilyen például a JavaBeans és JDBC (Java Database Connectivity) támogatás, valamint a biztonsági funkciók tökéletesítése. A mostani JDK már tartalmazza a SUN által a közelmúltban kibocsátott (JDK 1.1.2) biztonsági javításokat is. A működés sebessége is tovább nőtt, mintegy 30%-kal. A JDK 1.1 letölthető a http://service.boulder.ibm.com/jdk/s-feat.htm oldalról. *(A CD mellékletre az időközben kiadott végső változat került!)*

Grafikus programok jönnek és mennek **

Az előzetes híresztelésekkel ellentétben sajnos nem lesz OS/2-es változata a ColorWorks v3-nak. Nem kell azonban túlságosan szomorkodnunk, mivel a Photo>Graphics PRO v2 (lásd alább) mellett újabb grafikus program, az MD+F SX Paint jelent meg a piacon. A Modular Dreams Incorporated új termékét ingyen adják a néhány héttel ezelőtt bejelentett MD+F WebAK programmal, amelyet Web-grafikák készítésére fejlesztettek ki. Mindkét programról képek és további információ található a Modular Dreams honlapján: http://www.modulardreams.com. További jó hír, hogy a ColorWorks for OS/2 tulajdonosoknak árengedményt ad a Dadaware, amennyiben áttérnek a cég Embellish termékére. Az Embellish 2.02b béta verziója egyébként a közelmúltban jelent meg (ftp://ftp.dadaware.com/dadaware/embobeta.zip). *(CD mellékletünkön egy kicsit régebbi bemutató változat található.)*

ProNews/2 **

A Panacea Software (http://www.program.com/panacea) rövidesen új USENET-es hírcsoportolvasó-programmal fog megjelenni a piacon. A ProNews/2 béta 2 változatát már le is lehet tölteni a cég honlapjáról. *(Lásd a CD mellékleten!)*. Az ígéretek szerint a program sok, jelenleg unikumnak számító funkciót fog kínálni. Rövid ízelítő a kínálatból: Folyamatmenedzser (a sorban álló feladatok prioritásának változtatása, a feladatok felfüggesztése vagy újraindítása), a hírek hathatós boolean logikán alapuló szűrése, megadott cikkek figyelése és automatikus letöltése, intelligens dekódolás (a ProNews/2 önműködően megkeresi a több darabban kódolva küldött bináris anyagot, ellenőrzi az összes rész meglétét, majd letöltés után összeilleszti azokat), komplett szálak letöltése, valamint ROT13 támogatás.

WebExec **

Az IBM Software Server felhasználók számára lehet érdekes az IBM újonnan kibocsátott WebExec terméke, amellyel az IBM Software Serverek a hálózaton keresztül adminisztrálhatók. A Web

Execen kívül mindössze HTML 3.0 kompatíbilis, a Java és a JavaScript nyelveket támogató Web böngészőre van szükség (pl. Netscape Navigator for OS/2). A WebExec alapkiépítésben a következő kiszolgálókkal képes együttműködni: DB2 Database Server, Internet Connection Server, Lotus Domino Server, Transaction Server. A WebExec az IBM Software Server tulajdonosok számára ingyen letölthető a http://booksrv2.raleigh.ibm.com/eagle oldalról. Egy másik termékkel, a Web Administration program béta kiadásával (http://www.networking.ibm.com/cms/cs2dwn.html) pedig az IBM Communications Server (http://www.networking.ibm.com/cms/cs2abt.html) távadminisztrációja is lehetségessé válik.

Photo>Graphics PRO **

A kanadai érdekeltségű TrueSpectra cég (http://www.truespectra.com) forgalomba hozta képfeldolgozó programjának újabb verzióját. A cég termékismertetője szerint az új program nagyon sok újítást tartalmaz. Külön kiemelték a Web-grafikák készítéséhez használható funkciók nagymértékű bővítését. A Photo>Graphics 1996-os megjelenése óta folyamatosan a tíz legjobban keresett OS/2-es alkalmazás között van, s a 2.0-ás kiadás valószínűleg a viszonylag magas ár (250 USD) ellenére is jól fog fogyni. A jelenlegi Photo>Graphics tulajdonosok augusztus végéig vehetik meg az új verziót kedvezményes, 100 dolláros áron. *(A Photo>Graphics 1.04 Sampler változat megtalálható a CD mellékleten!)*

Pkzip 2.50 for OS/2 **

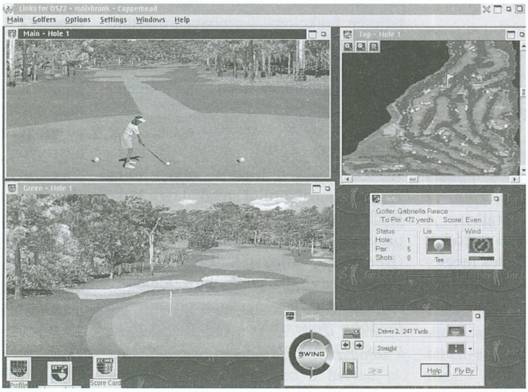
Igen hosszú szünet után adta ki a PKWARE (http://www.pkware.com) a Pkzip új, 2.50-es verzióját. A hírek szerint a Pkzipet intenzíven használó nagyvállalatok és bankok kérésére született meg a továbbfejlesztett változat. A Pkzipet regisztrálók számára egy "meglepetéssel" is kedveskedik a gyártó cég, mivel a tömörítőprogrammal ajándékba adja a PkzFind segédprogramot is, amely segítségével zip-archívumokban is lehet ezután keresni.

Golfjáték OS/2-re **

Újabb OS/2-es játékkal rukkolt ki a Stardock és az Access Software. Az eredetileg az Access Software által készített golfjátéknak az együttműködés eredményeként ezentúl OS/2-es változata is van: Links OS/2. Jó hír a játékot más operációs rendszer alatt is használók számára, hogy az OS/2-es változat által használt pályafájlok (*.CRH) teljesen kompatíbilisek a korábbi, más platformokra írt verziók fájljaival. Az érdeklődők fotókat láthatnak a Linksről a Stardock WWW lapján: http://www.stardock.com.

További játékáradat **

Az OS/2-es játékok kínálata tovább bővült a Stardock PlusPak részeként megjelent BUGS-zal, amely a hagyományos "lövöldözős" játékok sorát (pl. Galaga) folytatja. A játék lényeg az, hogy meg kell védenünk a Földet a világuralomra törekvő Bill ördög szörnyűséges űrhajóinak (Bill's Unbelievable Ghastly Spaceships, innen jön a BUGS) garázdálkodásaitól. *(Részletesebb leírást a CD mellékleten talál!)*



2. ábra: Golfozgat a kislány

Internet Adventurer 1.01 *

18 hónapig tartó fejlesztés és tesztelés után bocsátotta ki a dán érdekeltségű Adventure Software az Internet Adventurer első verzióját, amelyet rövid szünet után a javításokat tartalmazó 1.01 követett. A program tulajdonképpen igen gazdag internetes alkalmazásgyűjtemény. A shareware változat letölthető a http://www.inetadv.net oldalról. Ugyanitt bővebb információ és képek is találhatók a programról. *(Lásd a CD mellékleten!)*

PMINews: A Stardock PlusPak új tagja *

A Stardock (http://www.stardock.com) és a SouthSoft (http://www.southsoft.com) cégek együttműködésének eredményeként piacra került egy új OS/2-es hírcsoportolvasó program, a PMINews. Az OS/2 beépített NewsReader/2 programjából jogosan hiányolt funkciók nagy része a PMINews-ban már megtalálható. A fontosabb jellemzők között említhető a hírcsoport-kiszolgálók (és csoportlisták) párhuzamos támogatása, offline hírolvasási lehetőség, valamint bináris fájlok vidd-és-dobd-le technikával való dekódolása. A demó verzió a Stardock honlapjáról (http://www.stardock.com) tölthető le. *(Időközben megjelent a PMINews frissített, 1.01-es kiadása is. A programról bővebben is olvashat a következő számunkban!- A Szerk.)*

IPAD v0.2.5 - az intelligens vektoros rajzolóprogram *

Az IPAD egyike azon kevés rajzolóprogramoknak, amellyel vektorgrafikákat szerkeszthetünk OS/2 alatt. Sajnos az IPAD igazából DOS program, azonban van olyan verziója, amelyet teljesképernyős OS/2-szekcióra optimalizáltak. A tervek szerint az IPAD lesz a kezelőfelülete egy bonyolult, a közeljövőben elkészülő multiplatformos programcsomagnak. Az IPAD az intelligens jelzőt azzal a tulajdonságával érdemli ki, hogy a már létrehozott objektumok megpróbálják "kitalálni" a felhasználó gondolatait, s ennek megfelelően "viselkednek", ily módon a szerkesztés hatékonysága nagy mértékben megnő. A program letölthető az IPAD honlapról: http://www.demon.co.uk/titan. *(A szerző engedélyével rákerült a CD-re is!)*

RTF-ből IPF-be *

A jelenleg bétatesztelés alatt álló Rtf2Ipf segédprogram bizonyára nagyon hasznos lesz azok számára, akik gyakran

készítenek INF (információs) fájlokat. Az Rtf2Ipf ugyanis automatikusan alakítja át az RTF (Microsoft WinHelp Rich Text Format) formátumban készített anyagokat IPF nyelvre, amelyből aztán egy megfelelő fordítóval (IPFC) készül az INF fájl. A tesztverzió a http://www.backswc.com/rtf2ipf.htm oldalon található.

BBBS v3.33 *

A BBS-t futtatók és használók körében valószínűleg jól ismert a BBBS program, amelynek most jelent meg a 3.33-as kiadása. Az OS/2 mellett a program nagyon sok más operációs rendszeren (pl. Windows NT, Linux, PC-DOS) is fut. A BBBS azoknak is ajánlott, akik még csak most tervezik, hogy BBS-t nyitnak, mivel a csomagban gyakorlatilag mindent megtalálnak (pl. teljes Internet és FidoNet támogatás), amire szükségük lehet. A BBBS OS/2-es változata az ftp://ftp.bbbs.net/pub/dist/bbbs/bbbs_2.zip fájlban lelhető fel.

Newsbeat 1.00 *

A PMINews nem sokáig élvezhette a legújabb OS/2-es hírcsoportolvasóprogram címet, mivel néhány nappal kibocsátása után megjelent a már igen régen beharangozott Newsbeat is. A binary hírcsoportokat látogatók minden bizonnyal nagyon fognak örülni a Newsbeatnek, mivel integrálható a PMView-val és lehetővé teszi a kódolt állományok (uuencode) előzetes megtekintését is. A Newsbeat honlapja: http://www.cyberbeach.net/~minogue/newsbeat.html.

Navigator javítás *

A Netscape böngészőiben június közepén biztonsági hibát fedeztek fel, amely lehetővé tette, hogy a Web-kiszolgáló (JavaScript vagy Form oldalak böngészése közben) lokális fájlokat töltsön le a kliens gépről. A Netscape Corporation gyorsan elkészítette a javítást, amelyet a böngésző OS/2-es változatába is beépítettek. A javított böngésző (amely ezután azt is lehetővé teszi, hogy a programból négynél több példány is futhasson egyszerre) a szokásos helyről, a http://www.internet.ibm.com/browsers/netscape/warp oldalról tölthető le.

Mátrix - nem (csak) matematikusoknak *

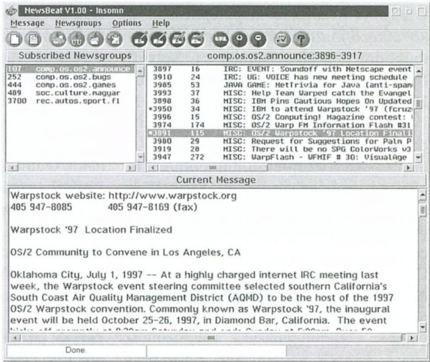
A CodeSmith Software fejlesztésének eredményeként jelent meg a Tetris-hez hasonló, Matrix nevű stratégiai játék. A BUGS-zal ellentétben a Matrix shareware és teljesen funkcionális próbaváltozat tölthető le a http://www.bmtmicro.com/catalog/matrix/matrix.html oldalról, illetve megtalálható a CD mellékletünkön.

LXOPT v1.22 *

Az OS/2 alá fejlesztők valószínűleg nagyon fognak örülni annak, hogy Mike Ruddy kiadta 32 bites kódoptimalizáló programjának teljesen ingyenes (freeware) változatát. Letölthető az ftp://hobbes.nmsu.edu/pub/os2/dev/util/lxopt122.zip fájlban. *(Sajnos a program további fejlesztése és támogatása megszűnt -A Szerk.)*

GammaTech IRC Fixpak *

A GtIrc 2.x tulajdonosok javítócsomagot tölthetnek le a GammaTech, Inc. honlapjáról (http://www.gt-online.com), amely az utóbbi hónapok során felfedezett problémákat orvosolja. A frissítés után az IRC ügyfél verziószáma 2.06-ra emelkedik.



1. ábra: NewsBeat 1.0: Mi újság?

Chorus/2 *

Hét nagyon hasznos, többségükben már régóta ismert és sokak által használt kiegészítő programot integrált egy csomagba a BMT Micro, Inc. A Chorus/2 kiadásának az volt a célja, hogy a felhasználók feltétlenül felfigyeljenek a "becsomagolt" alkalmazásokra: X-IT (Munkaasztal kiegészítő), DragText (a Drag&Drop technológia kiterjesztője), Smalled (gyors WYSIWYG szövegszerkesztő), SMEHTM Extensions (kiterjesztés a Smalledhez, mellyel az HTML szerkesztővé alakítható), Swap Monitor (tárcserefájl-figyelő), File Freedom (PM fájlmenedzser) valamint az előző hírünkben már említett Matrix játék. További információ: http://www.bmtmicro.com/chorus.

Kádár Zsolt

Decemberben megtudja!

December kilencedikén megjelenő számunkban nem kevesebb, mint 500 tippet és trükköt talál majd OS/2-höz! A hatalmas válogatás segítségével számos problémájára megoldást találhat anélkül, hogy szakértőhöz kellene fordulnia. Lesz benne szó mindenről:

- o telepítési tanácsok és trükkök
- o a Workplace Shell használata
- o teljes billentyűzet-kombináció lista
- o gyorsító és optimalizáló megoldások Warp 4-re és Warp 3-ra
- o tippek a munka gyorsítására
- o minták (template-ek) használata
- o CONFIG.SYS beállítások
- o ajánlott gépkonfigurációk

és még sok más!

A gyűjtemény INF formátumban, kényelmesen és hatékonyan kereshető formában készül majd el, így nem lesz szüksége a CD-re annak futtatásához!

Ha nem tud decemberig várni, akkor kezdje el olvasni a CD-n az OS/2 Kérdések és válaszok oldalakat és az OS/2 levelezési lista anyagát, ahol számos kérdésére már most választ kaphat!

Fixpak telepítése

Nem csak floppy-ról lehet FixPaket telepíteni, hanem közvetlenül a merevlemezről is. Ennek egyik módját mutatjuk be.

Az 1996-os decemberi számban (lásd a CD-ROM-on a 9612-fpk.htm fájlt) már szerepelt OS/2-es javítócsomagok (fixpakek) telepítése. Sajnos akkor nem sok szó esett a válaszfájlokról (response file), amelyekkel — többek között — a merevlemezről (vagy az OS/2 Times CD-ről :-) történő közvetlen telepítés is megoldható. Cikkemben ezt a hiányosságot szeretném pótolni. Mielőtt azonban belevágnánk, elevenítsük fel röviden a javítócsomagokkal kapcsolatos alapvető tudnivalókat!

Az IBM rendszeresen készít az OS/2 operációs rendszerhez javításokat, amelyeket többnyire koncentráltan, ún. javítócsomagok (FixPak) formájában bocsájt a felhasználók rendelkezésére. A tesztelések során megfelelőnek talált javítócsomagot nyilvánossá teszik. A telepítéshez szükség van még a telepítőprogramot (CSF = Corrective Service Facility) tartalmazó „kicker" lemezekre (ftp://service.boulder.ibm.com/ps/products/os2/fixes/wkickr) is. Nagyon fontos, az hogy az operációs rendszer verziójának és nyelvének megfelelő javítócsomagot és a „kicker" lemezek legújabb változatát töltsük le! Az IBM ftp kiszolgálóján a javítócsomagok és a „kicker" lemezek képfájlok (image file) formájában (*.DSK) találhatók meg. A letöltött képfájlokból a LOADDSKF.EXE programmal készíthetjük el a telepítőlemezeket, s az első „kicker" lemezen található SERVICE.EXE program indításával kezdhetjük meg a telepítést. A fixpakek (kumulatív jellegükből adódóan) az idők folyamán egyre csak nőnek, így telepítésükhöz egyre több mágneslemezre és időre van szükség *(Ez ügyben változás várható -A Szerk.)*. Mindkettő csökkenthető azonban, ha közvetlenül a merevlemezről telepítünk. Ez a módszer különösen akkor hasznos, ha központi CID (Controlled Installation and Distribution) kiszolgálóról akarjuk terjeszteni a javítókészletet, amelynek merevlemezét a frissítendő ügyfelek el tudják érni, hiszen ekkor csak egy helyre kell feltenni a fájlokat.

A merevlemezről történő telepítés első lépése az, hogy a javítócsomag fájljait tetszőlegesen megválasztott alkönyvtárba (pl. C:\FIXPAK) másoljuk a telepítőlemezekről (XCOPY A: C:\FIXPAK /S /E /V). A shareware cikkben említett Diunpack program segítségével a *.DSK fájlokból egyenesen a megfelelő alkönyvtárba másolhatjuk a fájlokat, és így elvileg egyetlen mágneslemezre sincs szükség a telepítéshez. Ugyanezt kell tennünk a második „kicker" lemezen található fájlokkal is. Az első „kicker" lemezre nincs is szükség.

Hogy a telepítés automatikusan történhessen, válaszfájlt kell készítenünk, amelyben „elmagyarázzuk" a telepítőprogramnak, mit is csináljon. Az alábbiakban tipikus válaszfájl-példát láthatunk, amelyre a továbbiakban C:\FIXPAK\FIXPAK.RSP néven fogok hivatkozni:

```
:FLAGS      REPLACE_PROTECTED REPLACE_
                 NEWER EXIT_WHEN_DONE
:SOURCE     C:\FIXPAK
:SERVICE
:SYSLEVEL \OS2\INSTALL\SYSLEVEL.OS2
:ARCHIVE    \ARCHIVE
:BACKUP     \BACKUP
:SERVICE
:SYSLEVEL \MMOS2\INSTALL\SYSLEVEL.MPM
:ARCHIVE    \ARCHIVE
:BACKUP     \BACKUP
```

A :FLAGS kulcsszó után szereplő opciók azt adják meg, hogy a telepítőprogram cserélje ki az írásvédett és az esetlegesen újabb dátummal szereplő fájlokat, valamint automatikusan térjen vissza a parancssorhoz a telepítés után. A :SOURCE-nál a fixpak fájlokat tartalmazó alkönyvtárt lehet beállítani. A :SERVICE kulcsszó adja meg, hogy telepíteni kívánjuk a javítócsomagot. Ha ezen a helyen pl. :BACKOUT szerepelne, akkor az a fixpak eltávolítását jelentené. Ugyanitt állhatna még a :COMMIT és a :REDIRECT kulcsszó is, amelyekről később lesz szó. A

```
:SYSLEVEL \OS2\INSTALL\SYSLEVEL.OS2
```

sor azt az információt hordozza, hogy a telepítőprogram a gépünkön lévő összes, a javítócsomaggal kompatíbilis OS/2-t szervizelje. Ha itt pl. a

```
:SYSLEVEL C:\OS2\INSTALL\SYSLEVEL.OS2
```

sor állna, akkor az azt jelentené, hogy csak a C:\-re telepített OS/2-re kívánjuk a fixpaket feltenni. Az

```
:ARCHIVE \ARCHIVE
```

sorban adjuk meg, hogy a telepítőprogram készítsen biztonsági másolatot az eredeti rendszerfájlokról, mielőtt felülírná azokat. Amennyiben nem először telepítünk fixpaket, akkor már létezik az \ARCHIVE alkönyvtár. Ekkor van szükség a

```
:BACKUP \BACKUP
```

sorra, amely azt közli a telepítőprogrammal, hogy a jelenleg használatban lévő, az előző fixpak által feltett rendszerfájlokat a \BACKUP alkönyvtárba mentse el. Ennek eredményeképpen az \ARCHIVE alkönyvtárban a rendszer telepítésekor, a \BACKUP alkönyvtárban pedig az előző fixpak által felrakott rendszerfájlok találhatók. A válaszfájl utolsó négy sorában a multimédia komponens szervizelését szabályozzuk az előzőekben leírtakhoz hasonlóan.

Ha készen vagyunk a válaszfájllal, akkor már csak annyi van hátra, hogy elindítsuk a telepítőprogramot. Ha ezt egy éppen működő rendszerből tesszük meg, akkor a telepítőprogram nem lesz képes az éppen használatban lévő rendszerfájlokat (pl. OS2KRNL) kicserélni, ezért ezeknek a fájloknak az újabb verzióját felmásolja egy ideiglenes alkönyvtárba, és beteszi a CONFIG. SYS-be az IBMCSFLK.EXE meghívását. Ez a program a következő rendszerindítás során (még mielőtt azok betöltődnének) ki fogja cserélni a problémás fájlokat. Az IBMCSFLK.EXE program az esetek többségében kifogástalanul működik, azonban ha igazán biztosra akarunk menni, akkor érdemes lemezekről (vagy másik partícióról) betöltött OS/2-ből indítani a telepítést, amelyet példánk esetében a következő paranccsal tehetünk meg:

```
C:\FIXPAK\FSERVICE.EXE
  /R:C:\FIXPAK\FIXPAK.RSP /S:C:\FIXPAK
```

A válaszfájlban megadott forráskönyvtárt felülbírálhatjuk az /S: opcióval. Lehetőség van arra is, hogy az alapesetben az \OS2\INSTALL alkönyvtárba készített naplófájlt (log) átnevezzük. A következő példa olyan telepítést indít el, amely a C:\FP26 alkönyvtárt tekinti a forrásnak, és ugyanide készíti a naplófájlt is, FP26.LOG név alatt:

```
C:\FIXPAK\FSERVICE.EXE /S:C:\FP26
               /R:C:\FIXPAK\FIXPAK.RSP
               /L1:C:\FP26\FP26.LOG
```

Ha el szeretnénk távolítani a javítócsomagot, akkor egy másik válaszfájlt

A CD mellékletünkről hasonló módszerrel lehet FixPaket telepíteni a \HTML\NETSCAPE\DRIVERS\drivers.htm fájlból

(pl. FPBKOUT.RSP) kell készítenünk:

```
:FLAGS     EXIT_WHEN_DONE
:LOGFILE   C:\OS2\INSTALL\SERVICE.LOG
:TARGET    ARCHIVE
:BACKOUT
:SYSLEVEL C:\OS2\INSTALL\SYSLEVEL.OS2
:BACKOUT
:SYSLEVEL
C:\MMOS2\INSTALL\SYSLEVEL.MPM
```

A megadott példában arra kérjük a telepítőprogramot, hogy a C:\-n lévő rendszer ARCHIVE alkönyvtárából (amely az előző példában C:\ARCHIVE volt) tegye vissza a telepítéskor felrakott fájlokat. Ha a :TARGET BACKUP sort adjuk meg, akkor a BACKUP (példánkban C:\BACKUP) alkönyvtárból tetetjük vissza az előző fixpak által feltett fájlokat. Ilyenkor az FSERVICE.EXE-t a következő módon kell indítani:

```
C:\FIXPAK\FSERVICE.EXE
/R:C:\FIXPAK\FPBKOUT.RSP /S:C:\FIXPAK
```

Ha elégedettek vagyunk a javítócsomaggal, akkor lehet a telepítőprogramtól COMMIT-ot kérni. A COMMIT letörli a BACKUP alkönyvtárt és eltávolítja a rámutató referenciákat. A COMMIT után tehát már nem lehet visszatérni az előző javítócsomag szintjére. Az ARCHIVE alkönyvtár mindig megmarad, vagyis a telepítéskori állapot elvileg mindig visszaállítható. A COMMIT-hoz a következő válaszfájlt (COMMIT.RSP) lehet használni:

```
:FLAGS     EXIT_WHEN_DONE
:LOGFILE   C:\OS2\INSTALL\SERVICE.LOG
:COMMIT
:SYSLEVEL C:\OS2\INSTALL\SYSLEVEL.OS2
:COMMIT
:SYSLEVEL C:\MMOS2\INSTALL\SYSLEVEL.MPM
```

A telepítőparancs pedig a következőképpen néz ki:

```
C:\FIXPAK\FSERVICE.EXE
/R:C:\FIXPAK\COMMIT.RSP /S:C:\FIXPAK
```

A válaszfájl tartalmazhatja még a :REDIRECT kulcsszót is. Ennek az a lényege, hogy a REDIRECT segítségével a telepítőprogramot rávehetjük másik ARCHIVE alkönyvtár használatára is. A CD-ROM-on található FIXPAK.TXT fájlban további válaszfájl-példákat találhatunk, valamit egy igen részletes javítócsomag-telepítési, visszarakási és elfogadási példát is. Zárásképpen pedig még egy fontos megjegyzés: a javítócsomag fájljainak (tehát a C:\FIXPAK alkönyvtár tartalmának) az előző javítókészletre történő esetleges visszaállásnál és a készlet elfogadásánál is rendelkezésre kell állnia!

Kádár Zsolt

Az FSERVICE.EXE paraméterezése

?	Segítségkérés.
/S:drive\path	A javítócsomag fájljainak elérési útvonala. Kötelező megadni!
/T:drive\path	A fixpak telepítéséhez használt OS/2-es rendszer elérési útvonala. Opcionális. Az FSERVICE ennek a rendszernek a BACKUP könyvtárából teszi vissza szükség esetén a CONFIG.SYS, STARTUP.CMD és AUTOEXEC.BAT fájlokat. Ha mágneslemezről indítjuk a rendszert a fixpak telepítéséhez, akkor az FSERVICE nem ellenőrzi az itt megadott útvonalat.
/R:path\file	A válaszfájl (teljes elérési útvonallal történő) megadása. Kötelező paraméter.
/L1:path\file /L:path\file	A naplófájl megadása. Nem kötelező használni.
/CID	Használatakor az FSERVICE a SEMAINT által készített biztonsági másolatból állítja vissza a következő fájlokat: o OS2BOOT o OS2KRNL o OS2LDR o OS2DASD.DMD o OS2SCSI.DMD o OS2LDR.MSG Opcionálisan megadható paraméter. Csak akkor használjuk, ha az FSERVICE SEMAINT által épített rendszerből fut!
/LARGE	Közöljük az FSERVICE-szel, hogy a fixpak CD-ROM-on található.
/B:boot-drive	A kapcsoló a könyvtárlista %BOOTDRIVE% paraméterének az értékét állítja be.

Diunpack

A Diunpack a savedskf program (lásd OS/2 Times 97/2 13. oldal) és az xdfcopy által készített fájlokat bontja ki. Ez eddig nem jelent különlegességet, ugyanis erre volt eddig is program: a loaddskf és az xdfcopy. Amiért a Diunpackra mégis érdemes szót vesztegetni az az, hogy a fájlokat nem floppyra, hanem tetszőleges alkönyvtárba csomagolja ki, így nem kell floppy keresgélésével, vagy virtuális floppy telepítésével vesződni. Elboldogul szinte minden formátummal (XDF, DSK — igaz régebbi DSK fájlokat nem hajlandó kezelni). Egyetlen hátránya az, hogy nem végez CRC ellenőrzést, azaz nem veszi észre az esetlegesen hibás fájlokat. A programot az IBM írta és semmiféle bővebb információt nem mellékeltek hozzá, így például azt sem lehet tudni, hogy szabadon terjeszthető-e vagy sem. Én természetesen az előbbire tippelek, mivel a Hobbes archívumban elérhető a fájl.

RXLS

Ez a kis program kötődik a hónap témája rovathoz, ugyanis REXX programokat fordít EXE fájllá. Nagy ördöngösséget nem csinál, ugyanis csak a REXX program kiterjesztett tulajdonságai (EA) elé tesz fejlécet. Az így „lefordított" program futtatásához továbbra is szükség lesz a telepített REXX környezetre. Akkor lehet hasznos, ha nagyobb programot alkottunk REXX-ben, amelyet pénzért vagy ingyen, de a források nélkül szeretnénk terjeszteni.

A program shareware, regisztrációs díja mindössze tíz dollár. Húsz dollárért lehet regisztrálni egy fejlettebb változatát is (RXCL), amely ún. REXX-X könyvtár készítésére alkalmas. Ezzel több (legfeljebb 50) REXX programot lehet egy fájlba fordítani, majd pedig a program nevét paraméterként megadva futtatni őket. Az RXLS-nek sajnos van egy kisebb hibája is: nem lehet meghajtót vagy alkönyvtárt váltani héj parancsokkal (pl. cd). Ezek helyett a

```
Call Directory('alkönyvtárnév')
```

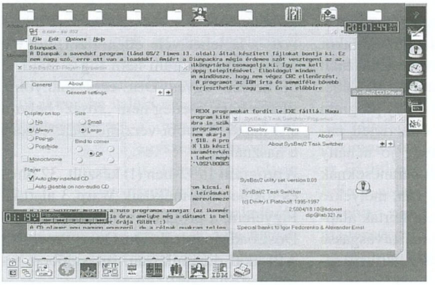
hívást kell használni a fordítandó REXX programokban.

SysBar

Ez tulajdonképpen nem is egy, hanem három kicsi program (1. ábra). Az egész csomag kibontva nincs 135KB. Ha hozzávesszük azt, hogy ez a 135KB a programok mellett a leírást is tartalmazza, akkor ez önmagában is értékelendő teljesítmény. A programoknak nem csak a merevlemezen, hanem futtatáskor a képernyőn elfoglalt helye is csekély. Lássuk tehát a három kis alkalmazást:

A *Task Switcher* mutatja a futó programok ikonját (az ikonméret tetszőlegesen állítható).

A *Clock* pici digitális óra, amelybe még a dátumot is belezsúfolták. A szerző szerint nagyon szépen elfér a WarpCenter órája felett :)



1. ábra: A SysBar/2 programjai

A *CD player* nagyon egyszerű, ugyanakkor a célnak sokszor teljes mértékben megfelelő CD lejátszó. Tetszetős és kicsi. Kizárólag CD lejátszására készült, nincsen benne „felesleges" egyéb funkció.

A három program nagyon hasonló kinézetű. Ez azzal magyarázható, hogy az általuk közösen használt kód egy külön DLL-be került helymegtakarítás céljából. A programoknál beállítható, hogy állandóan az előtérben legyenek-e, illetve hogy a képernyő egyik sarkához kötődjenek.

(A programnak nemrég nyílt meg a honlapja: http://www.lab321.ru/~dip/sysbar2/, a szerző ráadásul hamarosan számológép elkészítését is ígéri.)

Toronto Virtual File System (TVFS)

A TVFS az ebben a rovatban megszokott programoknál „nagyobb darab". Akik használtak már UNIX-ot, azoknak bizonyára hiányzik egy, a HPFS-nél rugalmasabb fájlrendszer. A TVFS igazából nem új fájlrendszer, hanem a meglevő (FAT, HPFS vagy egyéb) fájlrendszerek kibővítése. Telepítése egyszerű, csupán a CONFIG.SYS-be kell beírni a következő sort:

```
IFS=X:\TVFS\TVFS.IFS /LOGO
```

(Az X:\TVFS természetesen mindenkinek az az alkönyvtár legyen, ahová a program kicsomagoláskor került.)

Ezek után az X:\TVFS-t be kell írni a CONFIG.SYS LIBPATH és SET PATH soraiba is. A változtatások után a gépet újraindítva egy parancssorban ki kell adni a

```
TVCTL -w
```

parancsot. Ezek után lehet elkezdeni „mountolni". A „mountolás" annyit tesz, hogy egy gyökérkönyvtárba (ez UNIX gépeken a /, OS/2 alatt szabadon választott meghajtóbetűjel, pl. T:) tetszőleges alkönyvtárakat és fájlokat lehet felcsatolni. Például egy helyre lehet gyűjteni a segédprogramokat, a különböző alkönyvtárakban szerteszét heverő, az Internetről letöltött, de még át nem válogatott érdekes nevű könyvtárakat (saját tapasztalat :), vagy akár ftp kiszolgálóhoz lehet egy tetszőleges fastruktúrát készíteni a biztonság jegyében. (Ha ugyanis a potenciális betörő korlátlan jogokat is szerez az ftp kiszolgálón, akkor is ott van még a virtuális meghajtó, amelyről nem tudja elérni az igazi könyvtárakat.) Lehet egy partícionálás/programköltöztetés után linket (UNIX-ot használóknak ezt nem kell magyaráznom, akik viszont nem ismerik: az OS/2 "árnyék" (shadow) fogalmához hasonlít, de fájlrendszer szinten) létrehozni a program előző helyére, hogy pl. ne kelljen módosítani a konfigurációs fájlokat, vagy újratelepíteni a programot (linket csakis virtuális meghajtón lehet létrehozni!). Ennyi elmélet után lássuk végre a gyakorlatot! (Feltételezzük, hogy az OS/2 a D: meghajtóra van telepítve, amennyiben nem, úgy a D:\OS2 hivatkozást értelemszerűen írja át!) Adjuk ki bátran a

```
TVMOUNT T:
```

parancsot! Ezek után már van is egy T: meghajtónk, ami teljesen üres, de ez ellen mindjárt teszünk is valamit. Adjuk ki, hogy

```
TVLINK T:\OS4 D:\OS2
```

és már ott is ül a T: főkönyvtárában az

OS\2 alkönyvtár! Ide egyelőre nem lehet írni, de a

```
TVLINK -rw T:\OS_2 D:\OS2
```

parancs kiadása után máris változik a helyzet. Vigyázat, mert az innen letörölt fájlok a D:\OS2 alkönyvtárból is törlődnek (a UNIX-ban a link nem így viselkedik)!
Természetesen fájlt is lehet kapcsolni a TVLINK paranccsal:

```
TVLINK T:\KONFIG.SYS D:\CONFIG.SYS
```

A fenti példákban szándékosan „írtam el" az alkönyvtár illetve a fájl nevét, ezzel is mutatva, hogy ezeket tetszőleges név alatt fel tudunk csatolni.
A TVSCREEN paranccsal PM ablakot lehet megjeleníteni, amelyben a használt TVFS fájlok listája látható. Itt lehet megnézni, hogy használja-e egy program az adott fájlt. A fentebb ismertetett parancsok mellett létezik még jó néhány másik is, illetve ezeknek a parancsoknak sok hasznos kapcsolója is van, melyekre nem tértem ki. Akinek az itt leírtak alapján megtetszik a program, olvasgassa szorgalmasan a programhoz tartozó segítséget!

ShowDLLs

Valamelyik régebbi OS/2 Times számban ismertettem már a PMDLL programot. Nos, ez egy nagyon hasonló „kütyü", csak parancssoros kivitelben. A paraméterként megadott fájl fejlécében felsorolt DLL-eket írja ki. A programnak két fontosabb opciója van: a „/t", amellyel a használt DLL-eket fastruktúrában adja meg és felsorolja az egyes DLL-ek által használt további DLL-eket is, illetve a „/a", amellyel az összes használt DLL-t felsorolja (ugyanis a program a gyakran betöltött OS/2 DLL-eket (DOSCALLS.DLL, PMGPI.DLL stb.) alapesetben kihagyja a felsorolásból).
Azt is le lehet vele ellenőrizni, hogy a másolandó program nem fog-e majd az új helyén egyes DLL-eket hiányolni, illetve hogy esetleg nem „rossz" helyről származó, régebbi verziójú DLL-t akar-e használni.
A program 1992-ben (!) készült, 32 bites és tökéletesen működik Warp 3 alatt is. Figyelem, a program nem találja meg a DosLoadModule és DosQueryProcAddr API-kkal dinamikusan, futásidőben betöltött DLL-eket!

Which

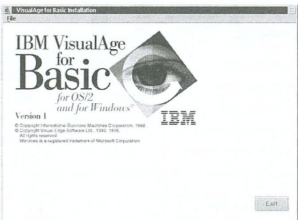
UNIX felhasználók számára nem ismeretlen a which parancs, amellyel tetszőleges fájl kereshető meg a PATH-ban. OS/2 alatt azonban bővebb funkciókkal is rendelkezik, ugyanis nem csak a PATH-t képes végigkeresni, hanem a BOOKSHELF-et, a DPATH-t, a HELP-et, a LIBPATH-t vagy akár tetszőlegesen megadható környezeti változó értékét is. Ezenfelül nagyon jól paraméterezhető az eredmény visszaadása is. Meg lehet adni például, hogy a program csak a visszatérési értéket adja meg (van-e találat vagy nincs), hogy a megtalált fájlok dátuma és méretét is kiírja, illetve hogy csak az első vagy az összes, a feltételnek megfelelő fájlt listázza ki.
Semmiféle utalást nem ad a program arról, hogy ingyenes-e vagy sem, én itt is az előbbire tippelek (a UNIX portokra általában ez a jellemző). Sajnos már nem tudom, hogy honnan származik, mivel a utility könyvtáramban bukkantam rá *(Az IBM AIX-like csomagjában is van hasonló program -A Szerk.)*. A program futtatásához szükség van az EMX könyvtárra.

Tóth Ferenc

VisualAge for BASIC

Szoftverfejlesztés mindenki számára, avagy Visual Basic IBM módra OS/2 alatt.

„Hurrá, van BASIC-ünk!" — szólhatna a hírmondó, meglátván az IBM VisualAge for BASIC-et. Való igaz, mindeddig hiányzott az OS/2 felhasználók készletéből egy igazán „átlagfelhasználós" nyelv, profi fejlesztőeszközzel. Bár a REXX ebből a szempontból megfelelő lenne (pl. Watcom REXX), de ebből hiányzott az átütő erő. Az IBM jelen esetben összekapta magát, és kihozta a BART kódnevű, sok helyen csak Visual Basic „gyilkos" néven emlegetett VisualAge családtagot. A csomagról nem kevesebbet állítanak, minthogy tökéletes többplatformos fejlesztőeszköz (OS/2 és Windows alá is fejleszthetünk vele), képes megenni a Visual Basic fájlokat (3.0-s verzióig, szöveges formátumban) és vezérlőket (VBX), lévén kódszinten kompatíbilis vele. Emellett könnyedén kezelhető, kitűnő az adatbázis és a hálózatos támogatása, meg egyébként is messze jobb mindenki másnál.



1. ábra: A VisualAge for BASIC logója

Valóban: sok helyen tökéletesen átgondoltnak, hovatovább megtervezettnek tűnik az eszköz, lévén azonban IBM termék, nem hiányzik belőle a Kék Óriás puritánsága. Ezek leginkább a segítségre és apró szolgáltatásokra terjednek ki, gyakran leginkább kényelmi szempontokból számítanak. (Például: miért nem képes a projektbe elmenteni a különböző ablakok helyzetét?) Az átgondolt objektumorientált felépítés sokat segít, a grafikus felület pedig nagyszerűen tervezhető és módosítható. A kódszétválasztás jót tesz az áttekinthetőségnek, de egy alaposabb BASIC tanfolyam beleférhetett volna a segítségbe. Mindent összevetve, a program jó, és egyedülállóan alacsony árával (25.000 Ft körül kapható a többplatformos, DB2 kiszolgáló támogatással rendelkező hálózatos verzió!) mindenképpen az „ajánlott vételek" közé tartozik.

Pudingpróba

A puding próbája az evés, tartja a mondás, így legegyszerűbb módja a magasztalt eszköz tesztjének egy félig-meddig valós alkalmazás létrehozása. Természetesen nem a járt ösvényen haladunk, azt vizsgáljuk meg, hogy egy „fejlesztésbe csöppent" felhasználó hogyan boldogul el a rendszerrel. Emiatt számos olyan problémát megemlítek majd, amely csak az első lépéseknél zavaró, és második vagy sokadik alkalommal már nem jelent problémát. (Például akkor, ha valaki rendesen elolvassa a dokumentációt, és egyszerűbb módszert is talál.) Így olyan sorrendben és elv alapján haladunk, ahogy az adja magát, nem úgy, ahogy a tervezők elképzelték. Szerencsére a fejlesztőeszköz kiállja ezt a próbát, ezért mindenképpen jó osztályzatot érdemel.

Első feladatunk egy apró alkalmazás létrehozása, amelyben egy alkönyvtár fájljaihoz fűzhetünk megjegyzéseket. Ezt a fajta kis alkalmazást gyakran használjuk a CD melléklet összeállításakor, vagy képaláírások mellékelésekor. Elsőként a munkát a legegyszerűbb résznél, a felület (az alkalmazásablak) létrehozásánál kezdjük. A bal oldalon található Toolbox egyes elemeire kattintva a Form1-en új vezérlőelemeket hozhatunk létre. A Toolbox elemei egyébként változóak, elhelyezkedésük függ attól is, hogy az alapmodulon kívül milyen készleteket importálunk még (pl. TCP/IP kommunikáció). Az első elem létrehozása után a bal egérgombbal a megfelelő kezdőpontra kattintva, majd a gomb lenyomva tartása mellett az elem jobb alsó mozgatva méretezhetjük és elhelyezhetjük azt. Jelen esetben először keretet (Frame1), fájllistát (File1) és szövegbeviteli mezőt (Text1) helyezünk el, valamint az elegancia kedvéért egy méretes gombbal (Command1) kitöltjük az üres helyet (Ízlések és pofonok —A Szerk.). A listában fog elhelyezkedni a jelenlegi alkönyvtárban található összes fájl neve, a beviteli mezőbe pedig a megjegyzéseket írjuk. A Command1 gombunk az alkalmazás lezárását jelenti majd.

Testreszabás

Következő lépésként általában érdemes finomhangolni a különböző gomb- és listaméreteket, mivel alapesetben a gridhez (rácshálóhoz) igazítja mindegyiknek a szélét. A Top (Tető) és a Left (Bal oldal) beállításokkal a bal felső sarok koordinátáit adhatjuk meg szinte minden objektum esetében. A Width (Szélesség) és a Height (Magasság) beállítások értelemszerűen az objektum megfelelő irányú kiterjedését jelentik majd. Egy-egy objektumot úgy tudunk kiválasztani, ha rákattintunk. Ilyenkor hajszálvékony keret jelenik meg nyolc pozícionáló négyzetecskével. Jó tanács: az F4 billentyűvel hívjuk le a Property Editor ablakot, és méretezzük a legkisebb szélességre és legnagyobb magasságra! Ezek után bármelyik objektumra kattintunk, annak tulajdonságai (színe, nagysága, kinézete és még sok egyéb) automatikusan megjelennek ebben az ablakban. Ha ez megvan, akkor visszatérhetünk méretezési feladatunkhoz.

(Jelen esetben ennek úgymond „nem sok teteje van", mivel a rácsra illesztett lista tökéletesen néz ki. A későbbiekben erre a műveletre egy-egy újabb objektum felhelyezésekor lesz majd szükségünk. Akkor már külön nem térünk vissza rá.)

Keressük meg a fent említett Top, Left, Width, Height beállításokat a Property Editor listájában (előtte mindig válasszuk ki a megfelelő objektumot!), majd kattintsunk a jobb szélen lévő adatmezők egyikére! Töltsük ki, majd nyomjunk Entert, vagy kattintsunk másik objektumra! (Ilyenkor rákérdez, hogy a le nem zárt változtatásokat elmentse-e, amit az OK-ra kattintva meg is tesz.) Ha eleget játszottunk alkalmazásunkkal, akkor lépjünk tovább!

Következő lépésként, ha elég erőt érzünk az alkalmazásunk elcsúfítására, nekiláthatunk az átszínezésnek. Ezt a Property Editorban a BackColor segítségével állíthatjuk be egy hexadecimális szám segítségével (az első &H karakterek után meg kell adni a Red, Green és Blue (azaz rendre vörös, zöld és kék)

összetevők értékét, majd még egy & következik), vagy a három pont által jelképezett ablakban kell kiválasztani a megfelelő színt. (Apró bosszankodás: a megszokott Warpos módszerrel, a drag'n'droppal rávontatott betűtípust és színeket az első futtatáskor elfelejti.) Itt említeném meg a szokásos többszörös kiválasztás módszerét, a megszokottnak megfelelően a CTRL billentyű lenyomása mellett több objektumot összefoghatunk. Azt, hogy mely objektumokat választottuk ki, a Property Editor felső listájában is követhetjük. Ilyenkor a közös tulajdonságok (mindegyiknek van pl. Captionje, azaz felirata, illetve háttérszíne és neve) jelennek meg. Azok előtt, amelyek nem egyformák (pl. más a színük), áthúzott egyenlőségjel áll. Ha valamelyik tulajdonságot ilyenkor átállítjuk, a listában szereplő összes objektum felveszi ezt az értéket. (Próbálkozzunk például a Font, Color és Caption tulajdonsággal! A többit lehetőleg hagyjuk békén!)

Minek nevezzelek?

A valódi programozás szempontjából döntő jelentőségű lesz a különböző objektumok elnevezése (Name). Ezt ezért valamilyen rendszer szerint kell megoldani, ráadásul úgy, hogy könnyen vissza tudjunk rá emlékezni. Én most azt a konvenciót használom, hogy elől következik az objektum neve, utána pedig esetleges típusa (pl. Kilepes Gomb). A Frame1, FileList1, Text1, Command1 elnevezésekből rendre Keret, FajlLista, Szovegbevitel, Kilepes Gomb lesz. (Sajnos a programozási nyelvek sajátosságai miatt mindenhol el kell hagyni az ékezeteket!)

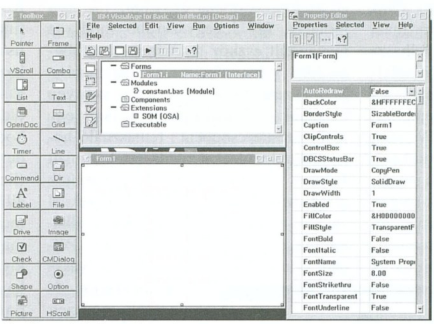
Ezek után majdnem készen is vagyunk, bár szeretnénk, ha a szövegbeviteli mező több sorból állna. Ennek megfelelően az objektum MultiLine tulajdonságát állítsuk igazra! Ezzel végeztünk is a grafikus felület megtervezésével.

Programozás

A fontosabb, és egyben nehezebb lépés a programkód írása, azaz a BASIC nyelv használata. Ehhez muszáj valamennyire művelődni, sajnos anélkül nem megy, bár kétségkívül könnyebb, mintha C-vel, vagy Pascallal kezdenénk. (Viszont nem is lehet vele olyan „hatékony" kódot készíteni -A Szerk.)

Feladatunk az, hogy a lista elemeire kattintva a hozzá tartozó megjegyzés megjelenjen a szövegmezőnkben, kilépéskor pedig a program mindezt mentse fájlba.

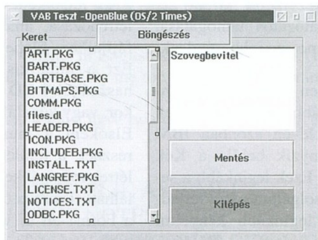
A Code Editor ablakát a megfelelő objektumra duplán kattintva hozhatjuk elő. Az Object/Class (objektum/osztály)



2. ábra: A fejlesztőrendszer

listában a megfelelő objektum neve áll, az Event/Method (esemény/metódus) mezőben pedig a megfelelő „történés", amelyre reagálunk. Nekünk elsőként a listára történő kattintásra kell kitalálnunk valamit. Ennek az eseménynek a neve „Click".

Itt kell megadnunk azt, hogy a kiválasztott elem megjegyzése bekerüljön a szövegmezőbe. A szövegmező tulajdonságaira a „Szovegbevitel." alapján hivatkozhatunk. Például a Szovegbevitel.Text adja meg azt, hogy mit is látunk a dobozban.



3. ábra: Valami ilyet szeretnénk

Először, az egyszerűség kedvéért írjuk bele a kiválasztott fájl nevét a Text tulajdonságba!

Ennél mi sem egyszerűbb: a Fajllista. FileName megadja a kiválasztott fájl nevét, amelyet azután a

```
Szovegbevitel.Text =
            Fajllista.FileName
```

sorral áttölthetünk.

A főablakban lévő nyílra (Play/Lejátszás gombra) kattintva elindíthatjuk programunkat. Ha nem vétettünk semmilyen hibát (ha elírtunk valamit, akkor arra a program figyelmeztet, megjelölve a hibás sort és leírva a hiba okát), akkor eltűnnek a projekt ablakai, és helyette megjelenik a mi saját programunk.

Lehet méretezni, kattinthatunk a fájllistában, és a szöveg önműködően megjelenik a mezőben. Ezek után módosítsuk kódunkat úgy, hogy a Kilépés gombra bezáródjon az alkalmazásunk:

```
Sub KilepesGomb_Click()
      Close
End Sub
```

Az első meglepetés akkor ér bennünket, amikor szeretnénk megnézni, hogy a Close parancs megfelelő-e számunkra az alkalmazásból történő kilépéshez. Nosza, nézzük meg, mit mond a Language Reference! „No linking found"! Magyarul: nem találja a megfelelő fájlt a BASIC segítsége. Semmi gond, parancssorból félműveltek lévén már tudjuk, hogy rajtunk nem fognak ki, annak ellenére, hogy a Language Reference a Visual Age for BASIC Information gyűjtőben sem szerepel. A TCP/IP csomag részét képező VIEWHELP parancs („VIEWHELP.EXE LR.HLP") azonban szintén nem készteti felbukkanásra a segítséget. Megjelenik ugyan a Window listben, de semmit se lehet vele tenni a bezáráson kívül (nem baj, azt megtesszük). Ekkor kezd az egyszeri programozó vallásossá válni, és mindenfélével próbálkozni. Ezek hosszú sorát nem írnám le, lényeg az, hogy egyszer csak működött a Language Reference! Ennek semmiféle racionális okát nem találtam, mindenesetre fogjuk arra, hogy a BASIC intelligens csatolója észrevette, hogy mennyire szükségünk van erre a fájlra, és kedvünkért belinkelte arra a helyre, ahol elsőre is meg kellett volna jelennie.

Most tehát a felsőbb hatalmak által megérintve újult erővel látunk neki munkánknak. Kiderül, miután megtaláltuk a Form objektum leírását (ha F1-et nyomunk úgy, hogy ki van jelölve egy ob-

jektum, akkor automatikusan a művele-tek, események és tulajdonságok leírása ugrik be), hogy az Unload paranccsal tudjuk lezárni. Programrészletünk így módosul:

```
Sub KilepesGomb_Click()
        Unload Form1
End Sub
```

Tárolás

Most következik a lényeg: el is kell tárolnunk ezeket a szövegeket. Jelenleg minden kattintás csak felülírja az előző szöveget, senki nem törődik a megőrzéssel. Hozzunk létre tehát egy tömböt, amelyben megfelelő számú szövegmező van!

A Code Editorban a Form modul General osztályában a Declarations részbe kell beírnunk a változódefiníciókat. Írjuk be:

```
Dim SzovegTomb() As String
```

Ezzel egy — jelenleg még nem meghatározható méretű — szövegelemekből (String) álló tömböt adtunk meg, amelyet majd futásidőben a ReDim utasítással adunk meg pontosan akkorának, ahány elem a fájllistában van:

```
Sub Form_Load()
 ReDim SzovegTomb(1 to Fajllista.ListCount)
End Sub
```

Ezek után meg kell oldanunk a tárolást. Erre legjobb pont a szövegbeviteli mező eltárolása (hiszen ha megváltozik, akkor érdemes tárolni), vagyis a Szovegbevitel_Change().

```
Sub Szovegbevitel_Change()
  If Fajllista.Listindex <> -1 then
        'vagyis van kiválasztott elem
    SzovegTomb(Fajllista.ListIndex+1) = Szovegbevitel.Text
  End if
End Sub
```

Ez persze még nem elég, fontos, hogy a szöveget mindig vissza is írjuk a mezőbe, így a Fajllista_Click() a következőképpen módosul:

```
Sub Fajllista_Click()
  If Fajllista.Listindex <> -1 then
    Szovegbevitel.Text = SzovegTomb(Fajllista.ListIndex+1)
  End If
End Sub
```

Itt az ideje futtatnunk a programot! Tökéletesen úgy működik, ahogy elterveztük, így már nincs más hátra, csak ki kell mentenünk egy fájlba az adatainkat.

```
Sub Form_Unload(Cancel As Integer)
  Open „files.dl" For Output As #1
  For i = 1 to Fajllista.ListCount
    Print #1, Fajllista.List(i-1)
    Print #1, SzovegTomb(i)
  Next
  Close #1
End Sub
```

Készen is van az első, és értelmes dolgot végző programunk!

Akinek van kedve, tovább finomíthatja a programot, hogy a kimenő lista formázottan jelenjen meg, vagy hogy ugyanebből a programból meg is tudjuk nézni a kimentett leírásokat (a files.dl fájl visszaolvasása után). Lássuk most ez utóbbi megvalósítását röviden!

Az ablakban lesz egy meghajtó-(MeghajtoLista) és egy alkönyvtárlista (KonyvtarLista), valamint egy szövegmező (SzovegLista), ahol előre megnézhetjük a files.dl fájl tartalmát, az ablakot pedig a SzerkesztesGombbal zárhatjuk le. A SzerkesztesGomb hatására az eredeti ablakban (Form1) az általunk kiválasztott könyvtárhoz tartozó files.dl fájlt szerkeszthetjük, vagy hozhatjuk létre.

Meghajtóváltáskor frissíteni kell az alkönyvtárak listáját, ha pedig lenyomjuk a SzerkesztesGombot, meg kell adnunk az eredeti ablak (Form1) fájllistájának az új alkönyvtárat, majd le kell zárnunk az ablakunkat.

Amikor kiválasztunk egy alkönyvtárat a Form2-ben, akkor onnan be kell hívni a megfelelő files.dl fájlt is megtekintésre, ha létezik ilyen.

A Form2-t a Form1-ben elhelyezett gombról indíthatjuk, vagy esetleg a fájllistán történő dupla kattintásra is behívhatjuk.

Problémát jelent, hogy az eredeti ablakban eddig kilépéskor mentettük el az információkat, most azonban egyszerre több fájlt is szerkeszthetünk, így a mentés funkcióit érdemes (és így kellett volna eredetileg is) külön gombra (MentesGomb) helyezni. Ilyenkor is csak akkor mentünk, ha szükséges. Ne felejtsük el, hogy mindig a teljes útvonalat kell már megadnunk, vagyis nem csak a „files.dl", hanem a Konyvtar Lista.Path &„\files.dl" sort!

Komolyabb, és nehezebben észrevehető probléma az, hogy eddig a program indulásakor meghatároztuk a szövegeket tároló tömb (SzovegTomb) méretét a ReDim paranccsal, azonban jelen esetben ezt már a Form1.FajlLista_Change eseményre (a lista változásakor más méretű tömb kell) újra végre kell hajtanunk.

Ezek alapján már csak a Form1-ben a fájl betöltése hiányzik. Itt valamivel komolyabb problémával találkozunk, hiszen nem biztos, hogy a kimentett files.dl minden fájlt tartalmaz a jelenlegi könyvtárban is! Ha új fájl kerül a listába, akkor a megjegyzések helyét csak a fájlnév alapján találhatjuk meg, nem tudjuk sorban „egy szuszra" beolvasni mindet. Ennek megfelelően először beolvassuk a fájlnevet, megkeressük a sorszámát az új listában, majd a megfelelő helyre a Szoveg Tombbe beírjuk a megjegyzést. Kimentéskor nem kell változtatni a stratégián.

Remélem mindenki kedvet kapott a próbálkozáshoz. A CD-n megtalálható az egyik helyes megoldás a fenti feladatra, illetve a fejlesztőkörnyezet teljes ismertetése. Egy angol szótárral felszerelkezve mindenki bátran nekiláthat a programozásnak!

Ambrózy Gábor

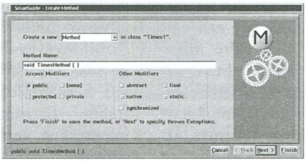
Figyelem!

Megfelelő számú jelentkező esetén következő számunkban VisualAge for BASIC tanfolyamot indítunk, hiszen az OS/2 Times CD melléklete révén mindenki rendelkezik a fejlesztőkörnyezet 60 napos próbaváltozatával. Amennyiben úgy érzi, mindig szívesen olvasná ezt a tanfolyamot, és használná is a benne leírtakat, hagyjon üzenetet a 221-0019 telefonszámon Ambrózy Gábornak, vagy írjon levelet az amby@eik.bme.hu elektronikus, illetve az Open Blue Bt. 1443 Bp. Pf. 256. postai címre!

VisualAge for Java

avagy „A programozás hatékony új perspektívája". Már nagyon régóta vártuk ezt a terméket OS/2 alá.

Az IBM igen régen meghirdette már, hogy számára a Java az egyik legfontosabb technológia. Míg a Microsoft azzal volt elfoglalva, hogy az MSN-t hirdesse, mint az Internet alternatíváját, majd kijelentse, hogy az Internetnek semmi értelme (aztán rá két hétre új részleget hozzon létre internetes fejlesztésekre), addig az IBM úton-útfélen a Java és a Network Computing szavakat kántálta mindenki fülébe. No igen, mondhatnánk, megint az IBM-nek volt több esze! Csakhogy az élet nem ilyen szép. Bár az IBM soha nem szűnt meg a varázsigét hirdetni, tettei azonban bizonyos területeken annál lassabban követték elhatározását. A versenytársak, köztük a Microsoft is, sorra jelentek meg a jobbnál-jobb fejlesztőeszközökkel, és a Kék Óriás úgy tűnt, megint lemarad valami jó dologról. Nemcsak, hogy nem csatlakozott a Java fejlesztőeszközöket forgalmazók tömegéhez, de többen felróttták neki, hogy a „támogatása" miatt a Sun nyelve, a Java, nem lesz képes megállni a saját lábán a Microsoft ActiveX-ével szemben.



1. ábra: Új metódus megadása

Szerencsére azonban végül megjelentek az IBM-es termékek is. Először a Network Station, az első működő NC implementáció, majd pedig sorra mások. Az egyik legfigyelemreméltóbb kezdeményezés, amely mindenki számára érdekes lehet, az Alphaworks site, a http://www.alphaworks.ibm.com címen. Itt folyamatosan új technológiákat mutatnak be, már a fejlesztés igen korai fázisában (nem béta, hanem alfa), hogy minél több véleményt hallhassanak egy-egy ötlet életképességéről. Ilyen pl. a Panoramix, a 360 fokos körképek böngészését lehetővé tévő kiegészítés, vagy a tartalom (forma) szerinti képkereső és osztályozó (nagyon jól működik!), vagy az azóta ismerté vált Bamba lejátszó, amelyet már korai fázisában is megismerhetett a szemfüles hálóböngésző.

Továbbra is adós maradt azonban az IBM két dologgal: az OS/2 beígért csodálatos Java támogatásával (a Warp 4 volt az első támogatott operációs rendszer, amely képes volt a hálóböngészőn kívül futtatni a Java programokat, mint „natív" alkalmazásokat, de utána megállt az élet), illetve egy jó fejlesztőeszközzel. A CEBIT-en azonban fellebbentették a fátylat a készülő eszközről, a VisualAge for Javaról. Kiderült az is, miért vártak annyit a fejlesztéssel: a lényeg a Java 1.1, amelyben számos új technológia (pl. a JavaBeans, nyomtatás) jelenik meg. Míg mások korán kihozták eszközeiket, addig az IBM megvárta, amíg a Java beérik az általa már kifejlesztett objektumtechnológiához. Amikor ez megtörtént, már csak át kellett szabni a VisualAge család környezetét a Java nyelvnek megfelelően. Persze a hosszas várakozás alatt sem tétlenkedett az IBM, átdolgozta az IDE-t (a mostani környezetet veszik majd át a VA család tagjai, pl. a VAC++ 4.0), és számos új funkciót beépített a VA legifjabb tagjába, mindezt a megújult logóval is jelezve.

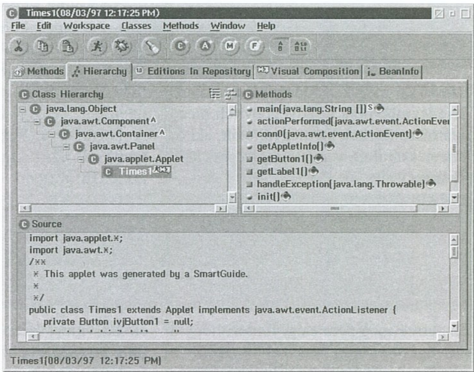
Az első meglepetés talán a telepítés: ez ugyanis nem a megszokott Software Installer programon keresztül zajlik. A második (és ez már kellemetlen) a gépigény: állítólag igen erős processzor és 64 MB memória szükségeltetik hozzá. Én egy P100, 32 MB RAM, Warp 4 konfigurációjú gépen teszteltem, és ez már tökéletesen megfelelt e célra. Sőt, meglepetésemre a Visual Buildernek megfelelő rész még gyorsabb is volt, mint pl. a VAC++ 3.0-ban (bár 32 MB memória alatt el sem indul a VAJ telepítőprogramja!). Komolyabb fejlesztéshez valószínűleg tényleg ajánlatos igen erős konfigurációt beszerezni.

Indítás után egy-egy SmartGuide script (ugyanaz a technológia, mint amelyet a Warp 4-ben a WarpGuide használ) vezet minket végig a fejlesztés első szakaszán. Választhatunk, hogy vizuálisan fejlesztünk, vagy egyből nekilátunk a kód megírásának. Ha a vizuális fejlesztést választjuk, megjelenik a Visual Builder, pontosabban a Composition Editor (CE).

Magáról a Composition Editorról nem szeretnék hosszasan értekezni, a 18. oldalon kezdődő VisualAge C++ cikk ugyanis bővebben foglalkozik vele. A CE-ben hozhatjuk létre a Java alkalmazás/kisalkalmazás felületét és a logika nagy részét, amelyet később néhány helyen a lényegi metódusokkal kell majd kiegészíteni. A project további részei a következőek:

A folyamatosan változó menüsor, illetve ikonsor mindig a megfelelő szerkesztőrészhez tartozó pontokat tartalmazza. Alatta található a több fülből álló lényegi projekt ablak, ahol a metódusokat (mezőket), az objektumhierarchiát, a különböző szerkesztett verziókat (beépített verziókövetés van a VAJ-ban), a Visual Compositiont (a már említett Composition Editort) és az úgynevezett BeanInfot érhetjük el.

A Methods részben magától értetődően alkalmazásunk/objektumunk metódusa-



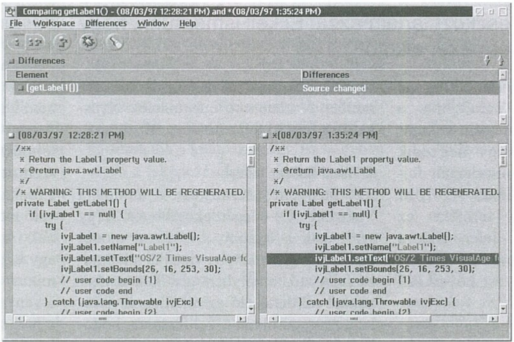
2. ábra: Objektumhierarchia metóduslistával körítve

CD mellékletünk \APPS\apps.inf fájljából telepíthető a VisualAge for Java Entry verziója

it módosíthatjuk. A felső részen lévő felsorolásban bármelyik sorra kattintva alatta egyből megjelenik a forrása (az előzőleg változtatott metódus forrása pedig elmentődik). Természetesen a forrás böngészésekor a VAJ használja a szintaxis-kiemelést, lévén ez már kötelező eleme mindegyik fejlesztőeszköznek. (Hozzáteszem még azt, hogy itt meg is lehet változtatni a színösszeállítást, míg ugyanez a próbálkozás a VAC++ esetén nem sikerült.) A metódusokat külön ablakban is megnyithatjuk és ilyenkor teljes ablakban szerkeszthetjük a forrást, illetve kezelhetjük a hozzájuk tartozó Repositoryt. Az Editions in Repository fülön minden esetben az adott objektum saját verziókövetése van, tehát ha megnyitunk egy metódust, akkor látjuk annak lebontott mentéseit is, nemcsak a teljes projectét. Ez nagyon fontos lehet olyankor, amikor egy programozó csapat dolgozik együtt, hiszen így könnyen kideríthető, ha lépésenként mindegyik modullal visszalépünk egy-egy szintet, hogy melyik új modul okozza a projekt adott fázisának működésképtelenségét. Létezik ezen kívül egy Repository Browser, ahol project, csomag és osztályszinten is böngészhetünk a különböző verziók között. Leghasznosabb azonban valószínűleg a Compare funkció: ennek segítségével automatikusan kideríthetjük, hogy két állapot között mi a különbség. Csak ki kell jelölni két állapotot, és a felbukkanó menüben a Compare menüpont segítségével máris megvizsgálhatjuk a forrást, akár egy „diff" programban. A két változat ráadásul szerkeszthető is ott helyben, nem kell visszatérnünk az Source editorba.

Új metódus a SmartGuide segítségével hozható létre; alapvető típusokat és kivételeket adhatunk meg, amely alapján a VAJ létrehozza a megfelelő fejléceket. (Sőt, még azért is szól, hogy kisbetűvel írjuk a metódusnevet, merthogy azt „úgy szokás".)

Létezik még egy érdekes lehetőség is, amelyik nem kapcsolódik szorosan a Methodshoz, de nagyon hasznos lehet: a Scrapbook. Ennek segítségével apróbb ötleteket jegyezhetünk fel, jegyzettömbszerűen. A különleges benne az, hogy egyes részeket kódként kijelölve futtathatunk, vagy megvizsgálhatunk anélkül, hogy külön projectet, vagy osztályt kellene létrehoznunk.



3. ábra: Áttekinthető verziókezelés

A Hierarchy önmagában nem sok érdekeset tartalmaz a szokásos Repository-hoz és más modulokhoz kapcsolódáson kívül. Az egyetlen igazán fontos funkció a menüben megjelenő Generate Javadoc, amely a megfelelő programozói dokumentációt készíti el ékes HTML nyelven. (A CD is tartalmaz ilyen dokumentációt, valamint az alapjául szolgáló kódot (Times1.java).) A hierarchiát egyébként fa és grafikus nézetben is megjeleníthetjük, ki-ki saját ízlése szerint. Ez a nézetváltás pl. az Editions in Repository résznél lesz majd érdekesebb, mivel ott előfordulhat, hogy egy vagy több lépcsőt visszalépünk, és onnan egy újabb ágon fejlesztünk tovább. Ilyenkor szerintem hasznosabb a grafikus nézet (Graph Layout).

A BeanInfot leginkább a Javaban már jobban elmélyedt fejlesztők értékelik majd, akik számos, leginkább nekik érdekes beállítást tehetnek meg itt. (Például mely beállítások tartoznak az Expert, vagyis szakértő körbe — ezek ugyanis alapesetben nem jelennek meg a Property Editorban.)

Néhány szó a Workbenchről, vagyis a munkaterületről: itt találunk minden fejlesztés alatt álló projektet, a különböző csomagokat, illetve azok metódusait. A Packages oldalon a metódusok között egyből megtaláljuk azokat is, amelyekkel a VAJ szerint gond van. Tulajdonképpen ugyanerről értesülhetünk, ha az Unresolved problems fülön keresztül nézelődünk, ott ugyanis számszerűleg és listázva is (magyarázattal együtt) megtaláljuk a problémákat.

A fenti rövid áttekintésből talán érezhető volt, hogy a Visual Age for Java az IBM egyik legjobban sikerült fejlesztőeszköze. Aki figyelmesen megnézte a képernyőfotókat, az azt is láthatta, hogy az IBM szokásával ellentétben nagyon komolyan odafigyelt az IDE kidolgozására. Olyannyira jól sikerült, hogy ha például a Visual Age for C++ csak ennyivel bővülne a 4.0-ás változatban, már akkor is megérné áttérni rá. A különböző részeket tucatnyi módon és logika szerint lehet elérni, az IDE valóban integrált, nagyon szoros egységet alkot, így csak annak leírása újságokat tudna megtölteni. Emellett a cikkemben mellőzött (de a 18. oldalon bővebben kifejtett) vizuális építő is egyedülálló megoldás. E két alapkő figyelembe vételével elmondható, hogy az egyik legjobb fejlesztőeszköz jelenleg a VisualAge for Java, amelynek Entry (dokumentáció nélküli) változatát CD-nken minden olvasónk megtalálhatja.

Ambrózy Gábor

Ami tetszett

- szép és (ennek ellenére) használható felület
- kitűnő vizuális fejlesztés (Composition Editor)
- jól tagolt munkakörnyezet
- kiváló verziókövetés minden téren
- jó hibakezelés
- megfelelő hibakereső módszerek

Ami nem:

- kissé nagy memóriaétvágy (32 MB-on rendszeresen 50 MB tárcserefájl)
- nem elég tiszta a különböző legenerálások rendje és hierarchiája, ezért veszett el fejlesztésem egy része

Éppen lapzártánk előtt jelent meg a Java 1.1.1 csomag. Megtalálható a CD \UPDATES\JAVA\111 alkönyvtárában

VisualAge C++ 3.0

A VisualAge programcsalád C++ tagja a komoly fejlesztéshez szükséges összes eszközt tartalmazza. El kell ismernünk, hogy a nagy kék kiváló terméket alkotott.

Manapság az objektumorientáltság az egyik legelterjedtebb paradigma a szoftverfejlesztésben. A magára valamit is adó szoftvercégek vagy negyedik generációs, vagy objektumorientált fejlesztőeszközöket használnak. Az IBM VisualAge C++ nevű terméke ez utóbbiak közé tartozik. Jelen cikk a csomag 3.0-ás verzióját mutatja be. A VisualAge C++ 3.0 a következő eszközöket tartalmazza:

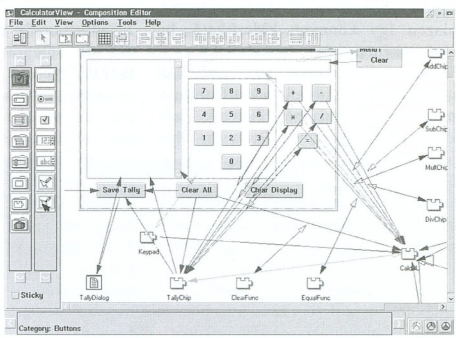
- C/C++ fordító
- Visual Builder (vizuális alkalmazásfejlesztő modul)
- Data Access Builder (adatbáziskezelő alkalmazások fejlesztését támogató eszköz)
- PM Debugger (hibakereső)
- LPEX szövegszerkesztő
- Performance Analyzer (teljesítményelemző segédeszköz)
- Object Utility/2 (Workplace Shell osztályok kezelését segítő eszköz)
- C++ Browser (C++ osztályhierarchia áttekintését segítő eszköz)
- számos további segédeszköz a fejlesztői munka megkönnyítéséhez

A fordító

A termék lelke a 32 bites C/C++ fordító. Számos opcióval rendelkezik, például képes különféle információkat elhelyezni a kódban, amelyeket más segédprogramok hasznosíthatnak (hibakereső, teljesítményelemző), beállítható, hogy 386-osra, 486-osra vagy Pentiumra optimalizált kódot generáljon, képes assembly nyelvre fordítani. Az egyik legérdekesebb lehetőség a DTS (Direct-To-SOM) beállítás, amely annyit jelent, hogy a fordító a C++ osztályokból SOM osztályokat állít elő (a SOM az OMG CORBA szabványának implementációja, nagyon leegyszerűsítve arra jó, hogy különböző objektumok képesek legyenek együttműködni függetlenül a hardver-szoftver platformoktól és az elhelyezkedésektől). Még rendkívül sok opció áll rendelkezésre, de azt hiszem, ezeket minden fejlesztőnek magának kell felfedeznie. A tapasztalataim szerint elég gyorsan fordít, 486-os gépen, 16 megabájt memóriával nem volt problémám a fordító sebességével. Az újság más részeiben részletesebb információk olvashatók a fordító teljesítményéről.

A Visual Builder modul

Véleményem szerint ez a modul a VisualAge C++ legérdekesebb része, így erről esik a legtöbb szó. Ez a modul egyfelől hagyományos vizuális alkalmazásfejlesztő, amelyben grafikus eszközökkel állítható elő az alkalmazás felülete, megspórolván ezzel rengeteg kódolást. Másfelől (és ez az igazán érdekes) a Visual Builderben lehetőség van az alkalmazás működésének grafikus eszközökkel való megadására is. Ennek egyik része az, hogy nem csak vizuális (azaz a képernyőn megjelenő) objektumok rajzolhatók fel a szerkesztőben, hanem háttérben működő (nem vizuális) objektumok is. Ezek lehetnek előre beépített, vagy saját készítésű komponensek is. Az alkalmazás működését az objektumok közti dinamikus kapcsolatok megadásával definiálhatjuk. Dinamikus kapcsolat alatt például az értendő, ha egy objektum egy adott esemény bekövetkezésének hatására üzenetet küld egy másik objektumnak. Ezzel szemben a statikus kapcsolatrendszer nem más, mint az osztályok öröklési, illetve tartalmazási hierarchiája (ez nem jelenik meg a szerkesztőben). Hatféle kapcsolatot különböztet meg a szerkesztő, ezek közül az egyik az előbb említett példa, az esemény-tagfüggvény kapcsolat. Más fejlesztőrendszerekben az említett példa úgy implementálható, hogy az adott objektumnak az adott eseményt kezelő metódusába beleírjuk a másik objektum megfelelő metódusát aktivizáló kódot. A Visual Builderben pedig csak egyszerűen össze kell kötni egy nyíllal a két objektumot, meg kell adni az eseményt és a metódust, és a Visual Builder legenerálja a C++ kódot. Ez a lehetőség magasan a piacon levő többi C++ alapú alkalmazásfejlesztő fölé emeli a VisualAge-et, legalábbis az én szememben. Azért nagyon jó ez a lehetőség, mert általános probléma az objektum-orientált szoftverfejlesztésben (is), hogy az elkészült szoftvertermék nem igazán felel meg a rendszertervnek. A rendszerterv objektummodellje egyértelműen tükröződik a C++ osztályszerkezetben, de a másik két modellnek, a funkcionális és a dinamikus modellnek való megfelelés már nem ilyen egyértelmű. A VisualAge-ben viszont kapunk egy eszközt, amely segítségével a dinamikus modell előírásai automatikusan érvényesíthetők a C++ kódban. (A dinamikus modell lényegében az objektumok közti lehetséges kölcsönhatásokat írja le. Akit érdekel az objektumorientált szoftvertervezés, annak ajánlom figyelmébe a László-Szirmay-Kondorosi szerzőhármas által írt, Objektum-orientált szoftverfejlesztés című könyvet.)



1. ábra: Composition Editor (Visual Builder)

Az általánosságok után térjünk rá a konkrétumokra! A Visual Buildernek háromféle nézete van, ezek: a Composition Editor (a fentiekben említett vizuális alkalmazásszerkesztő), a Class Editor és a Part Interface Editor.

A Class Editorban egy adott komponens információit adhatjuk meg, úgymint a generált fájlok neveit, a fejlesztő által írt header- és forrásfájlok neveit, a komponens ikonját a palettán (ha fel akarjuk venni a Composition Editor palettájára) stb. A Part Interface Editor egy ötoldalas jegyzettömb, ezeken az oldalakon adhatjuk meg a komponens attribútumait, metódusait és eseményeit. Az Attribute oldalon meghatározhatjuk az attribútumok nevét, típusát, beállító (set) és

lekérdező (get) metódusait és az attribútum értékének megváltozásakor bekövetkező eseményt. Az Event oldalon lényegében az adott komponens működése szempontjából jelentőséggel bíró eseményeket, ezek azonosítóit és paramétereit adhatjuk meg. Az Action oldalon definiálhatjuk a komponens metódusait, ezek visszatérési értékét és paramétereit. A másik két oldalon egyéb beállításokat tehetünk. Az itt megadott dolgoknak megfelelő deklarációkat generálja az eszköz, nekünk „csak" a függvénytörzseket kell megírnunk. Ezek a deklarációk a Class Editorban beállított fájlokba kerülnek. Ezek a fájlok nem íródnak felül generáláskor, hanem az eszköz hozzájuk írja az új deklarációkat. A Composition Editorban rajzolhatjuk meg az alkalmazást, illetve adhatjuk meg az egyes objektumok közti kapcsolatokat. A beépített komponenseket (például beviteli mező, jelölőnégyzet, rádiógomb stb.) palettáról választhatjuk ki. Lehetőség van saját osztályok felvételére is, amelyeket ugyanúgy használhatunk a kapcsolatok kialakításakor, mint a beépített összetevőket. A kapcsolatokhoz tartozó C++ kódot a Visual Builder generálja, így velük kapcsolatban semmiféle programozási feladatunk nincsen. A kapcsolatokhoz tartozó kódról érdekességképpen még annyit, hogy nem közvetlen a kapcsolat a kódban az objektumok között (tehát például nem az egyik objektum hívja meg közvetlenül a másik objektum megfelelő metódusát), hanem egy közvetítő objektum kapcsolja össze a résztvevőket. Ez azért jó, mert a kapcsolat törlésekor nem kell módosítani a résztvevőket, csak a közvetítő objektumot kell törölni.

Még néhány szó a teljesítményről: a VisualAge leginkább erőforrás-igényes része a Visual Builder. A tapasztalatok szerint legalább 486-os gép, és 16 megabájt memória kell neki, de a kényelmes használathoz nem árt 32 megabájt memória és egy Pentium. Hát igen, a minőségi eszközökhöz minőségi hardvert kell használni.

Adatbázis-kezelés

A legtöbb számítógépes alkalmazás kezel valamilyen adatbázist, éppen ezért a korszerű fejlesztőeszközöknek „kötelező" tartozéka az adatbázis-kezelő alkalmazások fejlesztését támogató modul. A VisualAge-ben ezt a szerepet a Data Access Builder, illetve az Open Class Library egyes osztályai töltik be. A következő adatbázis-kezelőkhöz történő kapcsolódáshoz van támogatás: DB2, Sybase és Oracle. A Data Access Builder segítségével valósíthatjuk meg a relációs adatbázistáblák és a C++, illetve SOM osztályok közti leképezést. A leképezés grafikus módon adható meg, ahol ikonok képviselik a táblákat és osztályokat, és kapcsolatok a leképezéseket. Ez nagyban segíti a leképezések áttekintését. Általában egy tábla egy osztályra képződik le, a tábla oszlopai pedig az osztály attribútumai lesznek. Ez a megközelítés akkor igazán hasznos, ha meglévő relációs adatokat kell kezelni VisualAge alkalmazásból. A Data Access Builder a leképezés megadása után legenerálja a megfelelő C++ (SOM) osztályokat, és biztosít metódusokat az adatok kezeléséhez, az Open Class Library pedig olyan osztályokat tartalmaz, amelyek az adatbázisokhoz történő kapcsolódáshoz, a tranzakciókezeléshez, illetve az adatmanipulációhoz biztosítanak műveleteket. Ezek absztrakt osztályok, a konkrét adatbázisokhoz való hozzáféréshez ezekből lehet leszármaztatni a konkrét osztályokat. A Data Access Builder által generált osztályok is ezek leszármazottai.

Debugger, Performance Analyzer

A programfejlesztés egyik alapvető eszköze a hibakereső, hacsak valaki nem tud kapásból hibamentes kódot írni. Mivel ez utóbbi igen-igen ritka :-), ezért mindenki jó hasznát veheti a PM Debuggernek. A hagyományos funkciók mellett a következő szolgáltatások állnak rendelkezésre:

- igény szerinti nyomkövetés (debug-on-demand): ha hiba lép fel az alkalmazás futtatása során, a debugger automatikusan elindulhat, és innentől lehet lépésenként futtatni az alkalmazást, kiíratni a változók vagy a verem tartalmát, stb.
- fejlett C++ nyomkövetési szolgáltatások, úgymint template-ek kezelése, osztályok megjelenítése és header fájlok nyomkövetése
- többszálas (multithreaded) alkalmazások nyomkövetése

A másik fontos segédeszköz a programok teljesítményének hangolását segítő Performance Analyzer. Ez az eszköz az adott program futtatása alatt egy fájlba gyűjt információkat. A futtatás után többféle grafikus formátumban lehet megjeleníteni az adatokat:

- Call Nesting diagram: a program futását függvényhívások sorozataként ábrázolja.
- Dynamic Call Graph: ezen az ábrán minden függvény csomópontként jelenik meg, és az ezeket összekötő nyilak mutatják a hívásokat.
- Execution Density diagram: az alkalmazás futását időszeletekre osztja, és megmutatja, hogy az egyes időszeletek hány százalékban vesznek részt az adott függvényben.
- Time Line diagram: az események időrendi kapcsolatait írja le.

Itt kell megemlítenem azt, hogy az optimalizálás nem azt jelenti, hogy trükkös programozói fogásokkal valamit gyorsítunk a programon, hanem azt, hogy jól választjuk ki az adatszerkezeteket és az algoritmusokat. Ennek alátámasztására álljon itt egy idézet Edward Youdontól, a strukturált szoftvertervezés egyik atyjától: „Könnyebb egy jól működő programot hatékonnyá tenni, mint egy hatékony programot jól működővé tenni." A Performance Analyzer jó eszközöket ad az ilyen szellemű optimalizáláshoz.

Néhány szó a VisualAge C++ 4-es verziójáról

Az eszköznek készül a 4-es verziója is (hamarosan várható belőle OS/2-es és Windows NT-s béta verzió), amelyben várhatóan a következő újítások lesznek:

- inkrementális C++ fordítás (azaz csak a változtatások által érintett függvények fordulnak újra, nem az egész kód)
- nem lesz szükség header fájlok, illetve make fájlok használatára (persze azért használhatók), a rendszer automatikusan sorrendezi a deklarációkat, és feltárja a függőségeket.
- a fordító támogatni fogja az új ANSI C++ szabványt, amely elsősorban a template-ek használatában hozott jelentős változásokat
- új IDE (Integrated Development Environment)
- Web-es alkalmazások fejlesztésének támogatása

Összességében annyit mondhatok, hogy nagyon tetszik a VisualAge C++, szerintem professzionális alkalmazásfejlesztő környezet. Elég nagy hátránya ugyan az igen magas erőforrásigény, de ha valakinek jó gépe van, akkor kipróbálni mindenképpen érdemes.

Kloknicer Imre

A CD \DEVEL\VACPP alkönyvtárában kiegészítések és forráskódok találhatók a VisualAge C++ termékhez

Borland C++ 2.0

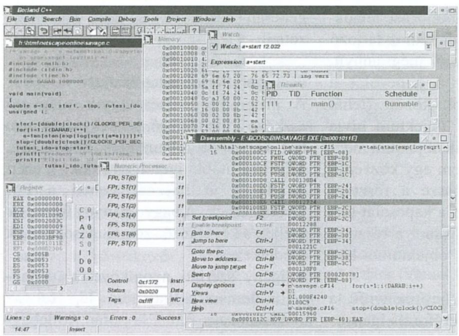
Az OS/2 alatti változat kinézetében nagyban hasonlít a Windows alatt futó társához, de tudásában sajnos messze elmarad tőle.

A Borland cég nagy hírnévre tett szert a fejlesztőrendszerek piacán. Forradalmi újításuk volt az IDE (Integrated Development Environment — egybeépített fejlesztői környezet), ahol nem csak a forráskódot lehetett szerkeszteni, de ugyanott lehetett elvégezni a fordítást és a hibakeresését is. A fejlesztőeszköz elkészült OS/2 alá, de később — úgy tűnik — abbahagyták fejlesztését. A Borland honlapon (http://www.borland.com) szinte semmi információ sincs a termékről, csak néhány régi és rövid dokumentum. A termék — általunk fellelt — utolsó OS/2 alatti verziója a 2.0, ezt teszteltük.

Valószínűleg nem okozok senkinek nagy meglepetést azzal, hogy a fejlesztőkörnyezet nagyban hasonlít a Windows alatti változathoz: a csomagban a Windows alatt megszokott programok OS/2 verziója van.

Az egész rendszer telepítve 70MB helyet foglal el a merevlemezen, de ebbe bele kell érteni a SMART nevű eszközt is, amellyel 16 bites Windows vagy OS/2, illetve Win32 forráskódot lehet áttenni 32 bites OS/2 kódra. Ehhez a program a jellemző API hívásokat, paramétereket, szimbólumokat stb. ülteti át. Azt azért senki se várja, hogy majd a SMART az egész programot átírja helyette, neki majd csak újra kell fordítania: a gépi átültetés után természetesen át kell nézni a kódot és kézzel javítani ott, ahol szükséges, a túl lassú kódot átírni kifejezetten OS/2 alá stb., mivel a SMART a portolást csak megkönnyíti. Aki csak új kódot szeretne fejleszteni, annak nem szükséges a SMART, amelynek törlésével közel 20MB helyet szabadíthat fel. A Borland C++ fájljainak lxlite-tal (*lásd OS/2 Times 97/2 szám*) történő tömörítésével további néhány megabájtot lehet nyerni a helyfoglaláson. A példák megközelítőleg 7MB helyet foglalnak, szükség esetén ezektől is meg lehet szabadulni.

A program indítása után az IDE jelenik meg, ahol egyes fájlokat is lehet szerkeszteni, de természetesen projekteket (komplex fejlesztési feladatokat) is képes kezelni. A szerkesztőben a syntax highlightnak (kulcsszavak, számok, változók stb. különböző színnel történő kiemelése) köszönhetően kényelmesen lehet a forrásfájlokat módosítani. Projektszinten be lehet állítani a fordító és linker opcióit, illetve egyéb beállításokat: például a fordított kód célját (DLL, EXE, illetve LIB), az INCLUDE és LIBRARY alkönyvtárait. A kívánt beállítások, illetve a forrásfájl elkészülte után következhet a fordítás, majd a szerkesztés (linkelés). Ezt egy lépésben is el lehet végezni a Make funkcióval (egyébként az IDE képes makefile-t készíteni a projekthez). A kész programot az IDE-ből is lehet futtatni, az esetleges hibás futás után pedig „kapásból" lehet debuggolni (1. ábra), persze csak akkor, ha fordítás előtt bekapcsoltuk a „Debug info in OBJs" kapcsolót. A debugger (nyomkövető) nem csak hogy nagyon kényelmesen használható, de rengeteg mindent képes megjeleníteni: visszafejtett kód, hívási verem, helyi változók, CPU/FPU regiszterek, szálak, memóriaterület stb. Mást nem nagyon érdemes róla írni, hiszen jól működik, és tudja mindazt, amit egy debuggertől el lehet várni.



1. ábra: Az IDE-be épített hibakereső

A Debuggerhez hasonlóan a Source Browser (forrásböngésző) is teljesen beépült az IDE-be , menüpontból lehet indítani (Search->Classes/Globals/Symbol at cursor). A Browser a program struktúráját grafikusan ábrázolja, így azt könnyen át lehet látni. Terebélyes C++ fájlok esetén jöhet jól, mikor csupán a forrásfájlt nézve már nehézkesen lehet megérteni, mi mit is csinál.

Egyedül a Workshop különül el az IDE-től, itt lehet az erőforrásokat szerkeszteni. A program ismeri az összes erőforrásformátumot, képes akár EXE fájlokból is kimazsolázni a képeket, ikonokat, párbeszédpaneleket stb., bár hajlamos néha „Unexpected file format" („Ismeretlen fájlformátum") üzenettel „megörvendeztetni" a felhasználót. Összességében azonban könnyen kezelhető, aki ismeri a Windows változatot, annak végképp nem fog nehézséget okozni.

A Borland C++ tartalmazza az Object Windows Library-t (OWL), amellyel könnyedén lehet PM alkalmazásokat fejleszteni. Ha azonban figyelembe veszszük, hogy a Borland cég nem fejleszti tovább a terméket OS/2 alatt, és a többi fejlesztőkörnyezet nem támogatja az OWL-t, akkor ez zsákutcának tűnik.

A csomag tartalmaz még assemblert (tasm), könyvtárkezelőt (tlib), erőforrásfordítót (brcc), ipf fordító (ipfc), make és még néhány egyéb hasznos segédprogramot. A segítség jónak mondható, részletesen tartalmaz minden fontosat, közel 13MB helyet foglalva el a merevlemezen.

A fejlesztőkörnyezet összességében kiforratlan, sajnos gyakran ad ki SYSxxxx üzeneteket, érződik rajta, hogy Windows port, a fordítóprogram sem a legjobban sikerült, túl sok optimalizálási beállítás bekapcsolásakor képes hibás kódot fordítani, amely SYS 3175 üzenet kíséretében elhal, de volt vele olyan „kalandom" is, hogy egy függvényre kijelentette, hogy túl bonyolult és nem tudja lefordítani. Ha a cél egy OWL-re alapuló Windows alkalmazás gyors portolás OS/2 alá, akkor hasznos lehet a Borland C++, de senki ne várjon elképesztő sebességet a fordított programtól. Amennyiben a Borland ismét nekiáll fejleszteni, akkor tudom ajánlani a fejlesztőeszközt, de addig inkább mindenki kerülje.

Tóth Ferenc

GNU C

Univerzális eszköz — ingyen! Ráadásul a GCC az egyik legjobb fordítóprogram OS/2 alatt.

Mint arról már az OS/2 Times 97/2-es számában is volt szó, a Free Software Foundation kereteiben működő GNU projekt célja, hogy szabaddá tegye a szoftverek áramlását. Ennek első lépése a saját, ingyenes, hordozható fejlesztőkörnyezet megteremtése volt. Eredmény: az egyik legjobb, legmodernebb C/C++ fordító, a GCC.

A program eredetileg nem OS/2-re született, de forráskódjának szabad elérhetősége miatt csak idő kérdése volt, hogy az OS/2 verzió is megjelenjék. Ma — a többi platformmal összhangban — a 2.7.2.1 verziónál tart a csomag, amelyre egyébként sok másik is épül (a GNU FORTRAN és PASCAL fordítója például az eredeti forrásból C kódot „gyárt", amelyet a GCC segítségével fordít le). Mivel profik írták, szinte minden olyan eszközzel rendelkezik, mint a kereskedelmi C fordítók — bár kevésbé látványos kivitelben.

A fordítón kívül a csomaghoz tartoznak még a szokásos C/C++ segédprogramok: a make nevű make program (újrafordítás esetén gondoskodik arról, hogy csak a legutóbbi fordítás óta megváltozott, illetve azoktól függő források forduljanak újra), a linker (ez UNIX hagyományok alapján a „beszédes" ld névre hallgat), a gdb nevű hibakereső (ha még valakit nem ijesztettek el az eddigi parancssoros programok, majd ez megteszi :-), assembler (gas), profiler (gprof) és a többiek. Természetesen rendelkezésre áll a szabvány C könyvtár, valamint egy általános célú (libg++) és I/O (libio) C++ osztálykönyvtár, tovább könnyítve a hordozható programok írását. A fordító a kevésbé felkapott Objective C nevű nyelvet is támogatja.

Fontos: OS/2 alatt a GCC-vel fordított, dinamikusan linkelt programok futtatásához szükség van az EMXRT könyvtárakra. Ezekről bővebben volt szó az OS/2 Times 97/2-es számában.

A telepítés

A telepítés elég egyszerű: az ftp://hobbes.nmsu.edu/pub/os2/dev/emx címről (vagy a CD mellékletről) be kell szerezni a csomag részét képező ZIP fájlokat, majd a kicsomagolás után a DOC könyvtárban található INSTALL.DOC fájlban leírtak szerint be kell állítani a környezeti változókat. Nagyon szerencsésnek tartom, hogy a GNU környezetben használt környezeti változók nevei nem ütköznek az egyéb C fordítók beállításaival (pl. az általánosan használt INCLUDE helyett a C_INCLUDE_PATH és CPLUS_INCLUDE_PATH változókat kell beállítani). Nálam kényelmesen megfér az IBM VisualAge C++ fordító mellett.

GCC

A rendszer alapja tehát a gcc néven futó C/C++ fordítóprogram (UNIX alatt a C++ fordítót g++ névvel illetik, OS/2 alatt ilyen megkülönböztetés nincs — a forrásfájl kiterjesztése és a kapcsolók döntenek a fordítási módról). A program gyorsan, jól optimalizált, megbízható kódot állít elő — ráadásul nem csak egy szoftver- és hardverarchitektúrán. A nyelvi változatok (dialektusok, szabványok), a fordítás menete, a kód előállítása mind-mind részletekbe menően állíthatóak (a teljesség igénye nélkül: az optimalizálás 24, a C++ dialektus 29 kapcsolón keresztül állítható). A fordító jól használható keresztplatformos fejlesztésre. A támogatott hardver platformcsaládok száma 17, az operációs rendszereket már numerikus túlcsordulástól tartva meg sem mertem számolni :-) Mindenesetre fordítottam és futtattam programot változtatás nélkül OS/2, Linux (x86), Solaris (UltraSparc), Digital UNIX (DEC Alpha) és AIX (RS/6000) alatt...

A fordító előkelő helyet foglal el az optimalizálás hatékonyságát értékelő listában (lásd külön cikkünk!).

GDB

Bár lehet, hogy többen idegenkedni fognak tőle, ez nagyon hasznos és a „tanulóidő" után kitűnően használható eszköz, amely túltesz nem egy kereskedelmi vetélytársán. Képes a kódot forrásszinten megjeleníteni, töréspontokat elhelyezni, a vermen oda-vissza lépkedni, változókat megjeleníteni, sőt, programozható is a DUAL nyelv segítségével. „Természetesen" támogatja a többszálú programok, DLL-ek és gyermekfolyamatok nyomkövetését is. És ami nagyon jól jön: a gcc-vel fordított programok memóriatartalma hiba esetén a core nevű fájlba kerül („coredump"). Ez az opció fordításkor kikapcsolható, hiszen adott esetben a core fájlba bekerülhetnek bizalmas adatok (jelszavak stb.) is, amelyeket jobb titokban tartani. A gdb természetesen támogatja a core fájl használatát, így pontosan meg lehet mondani, hogy mikor és mitől „halt meg" a program, bár az nem a hibakereső alatt futott (lásd „core-ból debugolni" :-).

A parancssoros és (kissé fapados) parancsorientált felülettől idegenkedőknek jó hír, hogy készül a PMGDB, amely PM „álarc" mögé rejti a „szálkákat".

GPROF

Az alkalmazás finomhangolására szolgál. A megfelelő opcióval fordított program futtatásakor létrejön egy fájl, amely megmondja, mikor „merre járt" a processzor, mely függvényben töltötte a legtöbb időt. Ebből a fájlból készít emberi fogyasztásra alkalmas kimenetet a profiler, amelyből kiolvasható az egyes függvényekben töltött idő (a teljes végrehajtási idő arányában illetve ms-ban) és a hívások száma.

LIBG++

A libg++ a GNU osztálykönyvtár. A benne található osztályok a következő feladatok megoldásában segítenek:
számok manipulációja (egész, racionális, fixpontos, komplex), karakterláncok kezelése, bitműveletek, véletlen számok előállítása, adatfeldolgozás (statisztika, hisztogramok), curses-alapú ablakozás, listák, láncolt listák, vektorok kezelése, halmazok, plexek, mapek, bagek, sorok és vermek készítése. Hátránya, hogy ezek egy része nem template-ek használatával történik, hanem karakterlánc-behelyettesítéssel „készül" a forrás, a genclass segédprogram használatával. Előnye viszont, hogy az így készített osztályokat más fordítókkal is meg lehet etetni (kis „reszelgetés" után).

OS/2 támogatás

Mint azt a bevezetőben is említettem, a GCC nem OS/2-n született. Ez persze nem jelenti azt, hogy meg kellene maradnunk a platformfüggetlen, karakteres módú alkalmazások írásánál: a GCC-

A CD mellékletünk \DEVEL\EMX alkönyvtárában megtalálható a legfrissebb teljes GCC fejlesztőkörnyezet

vel akár PM programot is lehet fordítani. Ezt az teszi lehetővé, hogy bár a GNU fordító az OS/2 rendszerhívásokat alapban nem tartalmazza, támogatja azok használatát, ha a megfelelő fájlok rendelkezésre állnak. Ezeket a „megfelelő fájlokat" hívják OS/2 Toolkitnek, és az IBM-től kell beszerezni (sajnos nem ingyen). A fordító nem támogat minden OS/2 API-t, nevezetesen a 16 bites függvények (amelyek szép számmal tenyésznek az OS/2-ben) meghívása nem lehetséges. Az OS/2-ben elterjedt többszálúság azonban nem okoz neki gondot, összes könyvtárának létezik egyszálas és többszálas verziója is.

Dokumentáció

A fejlesztőcsomaghoz jelentős mennyiségű, jó minőségű dokumentáció jár. Ezek GNU INFO formátumban érkeznek, de a gnudoc parancs és az ipfc segítségével könnyen OS/2 INF formátumúra alakíthatók. A segédprogramokról, a C könyvtár függvényeiről és az osztálykönyvtárról egyaránt bőséges tájékoztatást adnak a könyvek.

IDE helyett

Bár a GCC nem rendelkezik saját IDE felülettel, ez valamelyest pótolható. Magam a Boxer/2 nevű shareware szövegszerkesztőt használom (*lásd OS/2 Times 96/2*), amelyből a fordítóprogram meghívható, annak hiba- és figyelmeztető üzeneteit elfogja, majd egyenként előhozza. Alternatívaként az FTE nevű freeware-t és a Slickeditet ajánlom. Léteznek a GCC-re szabott kiegészítések az EPM szövegszerkesztőhöz is.

GCC hírek

Fejlesztés alatt áll az Intel Pentium processzorra optimalizáló GCC. A béta verziót a kísérletező megtalálhatják a CD mellékleten. Hamarosan várható a 2.8.0 verzió megjelenése.

Kovács István Attila

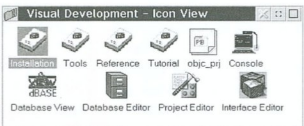
Visual Development

Vizuális Objective C fejlesztés OS/2 alatt

A Visual Development nevű shareware azok életét kívánja megkönnyíteni, akik a gcc fordító birtokában OS/2 alatti PM fejlesztésre adták a fejüket.

Telepítése egyszerű: a vd08.zip fájlt kibontva négy további ZIP fájlhoz jutunk, ezek rendre a csomag futtatható állományait, az OS/2 INF és Postscript formátumú dokumentációt, valamint a példaprogramokat tartalmazzák. A makeobj X (ahol X a telepítési meghajtó betűjele) parancs hatására létrejön a csomag gyűjtője a munkaasztalon. Ezután már csak a GCC megfelelő környezeti változóinak módosítására van szükség, és a gép újraindítása után kezdhetjük az ismerkedést.

Az első meglepetés: a program az Objective C nyelvet támogatja. (Ez engem kissé elkedvtelenített, mert nem beszélem ezt a nyelvet.) Nagy bátran nyakon ragadtam a program gyűjtőjében lévő mintát, és a Munkaasztalra ejtettem. Létrejött egy új gyűjtő, amelyben egy előkészített, fejlesztésre kész projekt várt.



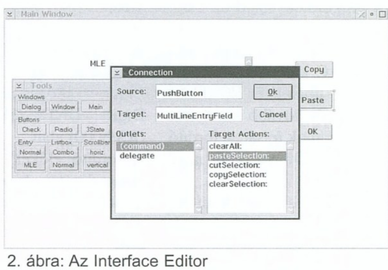
1. ábra: A program gyűjtője

A Project Editor

A projectfájlra (pb.prj) kattintva megjelent a Project Editor (gyakorlatilag egy jegyzettömb). Ez sajnos nem vette figyelembe, hogy nagy felbontásban használom a videokártyát, így rögtön át is kellett méreteznem az ablakot, amit aztán minden indításnál megismételhettem, mert az ablak nem volt hajlandó megjegyezni a beállított méretet. Ennél még jobban zavart, hogy a jegyzettömb alján látható fülek a lapozónyilak hatására pont fordítva mozognak, mint várnám.

A Project Editorban állíthatóak be a fordítást szabályozó opciók: a felhasznált könyvtárak, a futtatható célfájl neve, a felhasználói felületet megadó, az osztályokat megvalósító, illetve felületüket definiáló fájlok, és az egyes fájltípusokhoz használandó szerkesztőprogramok.

Már létező fájlok vontatással tehetők a megfelelő oldalakra. Új osztályok létrehozására a File menü Edit Classes parancsa szolgál, itt adható meg az osztály alapját adó összes adat: az osztály neve, szülője stb. Az itt megadott információ alapján az osztály forráskódjának vázát a program önműködően elkészíti, amiről az Objects fület kiválasztva meg is győződhetünk.



2. ábra: Az Interface Editor

Az Interface Editor

Az Interface Editor a program felhasználói felületének megtervezésére, valamint — ami jobban indokolja a program nevében szereplő „visual" jelzőt — az objektumok közötti kölcsönhatások vizuális megadására szolgál. A rendelkezésre álló, előre definiált osztályok, a vezérlőelemek (ablaktípusok, gombok és listák), valamint az általunk definiált osztályok használhatók, az IBM Visual Builderéhez hasonlóan.

Osztálykönyvtárak

A csomag igen fontos részét képezi három osztálykönyvtár: általános célú (fájlkezelés, named pipe-ok használata és hasonlók), valamint PM alkalmazások fejlesztését, illetve DBase III fájlok kezelését megkönnyítő segédosztályok. Ezek mellett természetesen használható minden, a GNU fordítóhoz járó könyvtár is, igen jól használható, a feladatok széles körét lefedő eszközkészletet biztosítva.

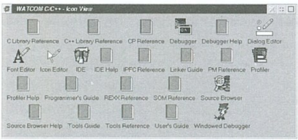
Összefoglalásként annyit mondanék, hogy a programon látszik: ígéretes, de jelenleg bizony béta program. A dokumentáció és a program egyaránt kiforratlan, komoly fejlesztése még nem ajánlom.

Kovács István Attila

Watcom C++

Bár sok OS/2 alatti C++ fejlesztőkörnyezet létezik, a Watcom C++ mégis kiemelkedik közülük a nagyon jó kódot készítő fordítójával és a könnyen használható kiegészítő programjaival.

A Watcom cég 1994-ben csatlakozott a Powersoft Corporationhöz, majd egy évvel később a Powersoft és a Sybase, Inc. egyesült, aminek köszönhetően a Sybase a hatodik legnagyobb szoftvercéggé vált. Természetesen a Watcom cég



1. ábra: A gyűjtőben minden egyhelyütt

C++ fejlesztőkörnyezetének fejlesztése eközben nem állt le. Nemrég adták ki a 11.0-s verziót, amely — többek között — sok Windows alatti újdonságot tartalmaz. Az OS/2 alá fejlesztőknek sem kell elkeseredniük, hiszen a Watcom C++ OS/2 támogatása mindig is jónak volt mondható. Szerkesztőségünkben még a 10.0-s verziót vettük közelebbről szemügyre.

A telepítés egyszerű volt: ki kellett jelölni, hogy milyen platformon akarunk milyen platformra fejleszteni (!) és meg kellett adni néhány további beállítást. Telepítés után a teljes OS/2 fejlesztőkörnyezet (OS/2 alatt 32 bites OS/2 alkalmazások fejlesztése) 46MB helyet foglal HPFS partíción úgy, hogy ebben benne van minden eszköz: példaprogramok, IPFC, debugger, profiler, IDE, fordító, részletes segítség, SOM fejlesztőkészlet stb. Amennyiben a „kevésbé fontos" dolgoktól megszabadulunk, azaz a példaprogramokat és a SOM fejlesztőkészletet elhagyjuk, akkor kb. 30MB hely elég a programnak, ami igazán nem mondható soknak. Ha a segítségfájlokat a CD-n hagyjuk, akkor további hely, mintegy 11MB takarítható meg. A telepítés után a Munkaasztalon létrejön a Watcom C++ gyűjtő (1. ábra). Ha az ikonok valamilyen ok miatt — például az OS2.INI fájl megsérülése — eltűnnének, akkor a telepítőprogramot ismét futtatva be lehet állítani, hogy csak a gyűjtőt hozza létre.

A termék néhány jól elkülöníthető részből áll: IDE, azaz az integrált fejlesztőkörnyezet, a PM és az ablakos hibakereső, a profiler, a browser, az IBM Developer's Toolkit for OS/2 2.1 és a segítségfájlok. Nézzük ezeket sorban!

IDE

Az IDE-ben (2. ábra) lehet magát a fejlesztést végezni. Első lépésként egy projektet kell létrehozni, ahol meg kell adni a fejlesztés célját (EXE — parancssoros, PM, teljesképernyős, virtuális eszközmeghajtó, HLP segítségfájl, DLL vagy LIB), majd meg kell adni a szükséges forrásfájlokat, de lehetőség van már előzőleg megint fájlok beemelésére is. A forrásfájlra duplán kattintva elindul az EPM, amellyel a forrásfájlt lehet szerkeszteni. Sajnos sehol nem találtam beállítást alternatív szerkesztőprogramhoz. Miután elkészült a mű, projekt-, illetve fájlszinten be lehet állítani a fordító kapcsolóit (optimalizálás, hibakeresési információ, kódgenerálás stb.). Nagyon jó ötletnek tartom, hogy minden opció mellett ott található a parancssorban megadható, vele egyenértékű kapcsoló is. A kapcsolók beállítása után elkészíthető a program (Sources->Make vagy F3). Hibás kód esetén az alsó ablakban (IDE Log) jelenik meg a hibaüzenet, amely pontosan megmondja, mely fájl melyik sora nem értelmezhető. Erre duplán kattintva megnyílik az EPM, a kurzor a hibás sorra kerül, amelyet még kiemeléssel is szemléltet. A hibát javítva újra kell fordítani a programot, aztán lehet büszkén futtatni. Természetesen első nekifutásra a program ritkán működik tökéletesen, ekkor jön jól a hibakereső. A debugger sincs integrálva az IDE-be, amely nevével ellentétben inkább csak keretet ad a fejlesztési munkának és adott opcióra a megfelelő külső programot hívja meg.

Debugger

A Watcom C++ debuggere (3. ábra) az egyik legjobb hibakereső, amellyel valaha is találkoztam. Nagyon kényelmes, rengeteg mindent lehet benne

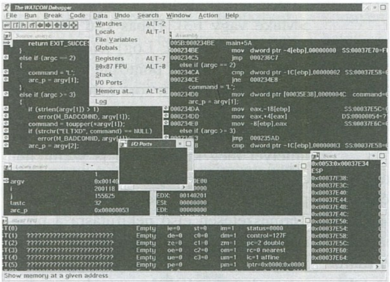
A C fordítóprogramról még a 23. oldalon levő C Fejlesztőkörnyezetek összevetése cikkben található információt. A részletes teszteredményeket a CD melléklet tartalmazza.

* * *

A CD-re helyhiány miatt nem kerültek fel javítások a Watcom C++ termékhez, de igény esetén a következő lemezen megjelennek. Kérjük tehát a Kedves Olvasót, írja meg, melyik verziójú Watcom C++ termékhez érdekelné javítás!

A Watcom C++ honlapja: http://www.powersoft.com/products/languages/watccpl.html

egyszerre megjeleníteni: CPU/FPU regiszterei, I/O portok, a verem, szálak, helyi/globális változók, a program C utasításainak assembly megfelelői, watches (azaz a figyelt változók/kifejezések) stb. A debugger a hasonló prog-



3. ábra: Aki ezzel sem találja meg a hibát...

ramoktól megszokott funkciókat nyújtja: lehet utasításonként lépkedni, töréspontokat, feltételes töréspontokat elhelyezni a programban, lehet a kódot a kurzorig végrehajtani, visszajátszani (replay), többszintű undo/redo stb. A program menüvezérelt karakteres módú verziója is megtalálható a csomagban, amellyel természetesen mindazt el lehet érni, amit a PM felületű változattal. Még megemlíteném, hogy a debugger bejelentkező logója nagyon ötletes: rovargyűjtemény (a bug szó eredetileg poloskát jelent) látható rajta olyan megnevezésekkel, mint Nullus Pointerassignment, Stackaphalia Overflowtian. :)

Profiler

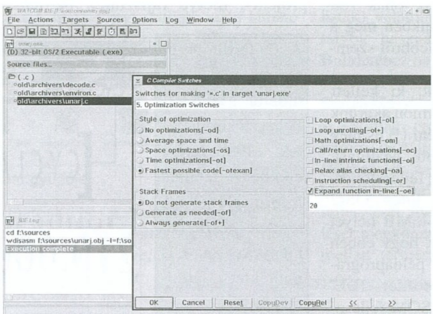
A Watcom C++ profilere elég egyszerű, de a célnak nagyon jól megfelelő eszköz. Használata előtt le kell futtatni a programot az IDE-ből a Targets menü Sample pontjával, amely futás közben statisztikát készít a programról, majd meg kell hívni a profilert (Targets->Profiler). A program az elkészült statisztikát jeleníti meg modulokra lebontva. Természetesen ez a bontás finomítható előbb C utasításokra, majd akár assembly utasításokra is. A program képes megjeleníteni az abszolút és relatív eltöltött időt. (Utóbbi az aktuális kódrészletben az adott utasítással eltöltött időt mutatja a teljes kódrészlet (pl. függvény vagy C utasítás) százalékban.) Az itt megjelenített statisztika alapján lehet a kritikus programrészletet azonosítani, majd azt módosítva a programot felgyorsítani.

Source Browser

A Source Browserrel többek között C++ források objektumainak kapcsolatát, az osztályöröklődés hierarchiáját lehet megjeleníteni. Ennek segítségével bonyolult, akár mások által írt objektumorientált kódot is gyorsan át lehet látni, fel lehet mérni, hogy milyen osztályok vannak benne, melyik miből származik, sőt tetszőleges osztály definícióját meg lehet jeleníteni dupla egérkattintással. A Source Browser használatához fordítás előtt az IDE-ben be kell kapcsolni az „Emit Browser info” opciót (parancssorban -db).

IBM Developer's Toolkit for OS/2 2.1

Igaz, már kicsit régi, de még mindig hasznos programgyűjtemény. Található benne többek között IPFC a segítség-

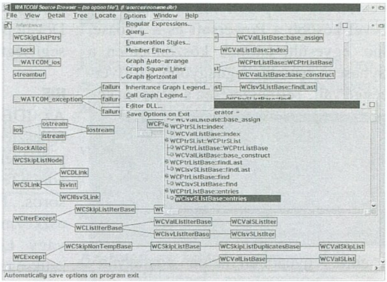


2. ábra: Optimalizálási beállítások az IDE-ben

fájlok (HLP és INF) fordításához, Resource Compiler (erőforrásfordító) az erőforrásoknak EXE/ DLL fájlokba történő fordításához, Dialog Editor (párbeszédablak-szerkesztő) a PM alkalmazások ablakainak megrajzolásához, Font Editor (betűkészlet-szerkesztő) FNT fájlok módosításához, az OS/2-ből ismert Ikonszerkesztő, Executable File Header Utility a végrehajtható programok fejlécének megjelenítéséhez és módosításához. Külön INF fájl ismerteti az eszközkészlet segédprogramjait, azok használatát és beállításait.

Segítségfájlok

Igencsak bőséges mennyiségű segítségfájl jár a termékkel. Bár az nem mondható el, hogy a mélyen a részletekbe merülnének, de sok hasznos információt tartalmaznak. Megtalálható itt többek között felhasználói kézikönyv, a C/C++ függvények és PM API-k leírása, SOM programozásról szóló információ, IPF



4. ábra: A Source Browser

fájlformátum ismertetése és egy REXX leírás, amely ismerteti, hogyan lehet REXX függvényeket meghívni C programból, és fordítva, azaz hogyan lehet C függvényekkel bővíteni a REXX programokat. Ezek mellett sok parancssorból hívható program is található. Természetesen található a csomagban make (az IDE a projectekhez automatikusan makefile-t készít, amelyet így parancssorból is könnyű fordítani), assembler és disassembler, sok kisebb segédprogram (pl. touch, library manager) de a linker és a fordító is hívható parancssorból.

Összegzés

A fejlesztőrendszer nagyon jól kidolgozott, komoly keresztplatformos fejlesztésre is alkalmas, nem utolsósorban azért, mert a csomag tartalmazza a DOS, Windows, OS/2, Netware és AutoCAD alatti fejlesztéshez szükséges eszközöket is. A rendszer memóriakövetelménye egyáltalán nem nagy, 16MB-s gépen már már egészen jól lehet vele fejleszteni, ráadásul a fordító elég hamar elkészíti az egyébként igen jó kódot.

Tóth Ferenc

C fejlesztőkörnyezetek összehasonlítása

A C nyelv hajlott kora :-) ellenére ma is az egyik legjelentősebb nyelv a számítástechnikában, ezért fontosnak éreztük a bemutatott C környezetek összehasonlítását.

Fordítók

Három különböző tesztprogramot fordítottunk le mindhárom fordítóval, és összevetettük az így kapott programok gyorsaságát és pontosságát. A részletes teszteredmények és a tesztprogramok forrásfájljai a CD mellékleten találhatók. Az eredmények alapján nehezen lehet egyértelmű erősorrendet felállítani a fordítók között, ráadásul a fordító a fejlesztőkörnyezetnek csak egyik, bár kétségkívül kulcsfontosságú része. Az azonban nem lehet véletlen, hogy a Watcom cég C fordítója mindhárom tesztben elsőként végzett nem csak sebességben, hanem pontosságban is. Saját tapasztalataim is nagyon jók a Watcom termékével. Másodiknak talán a gcc-t lehet nevezni, bár a tesztekben nem szerepelt túl jól, de nagyon jó tapasztalatok vannak vele. Harmadik és negyedik helyet megosztva a Borland és az IBM fordítója kaphatja. A teszteredmények a Borland fordítóját jobb helyre jogosítanák fel, de komoly gondok vannak vele: néha összetett optimalizáláskor hibás kódot generál. Az IBM fordítója számomra meglepetésként került ilyen hátra, mivel véleményem szerint ez a fejlesztőkörnyezet a legjobb, és mellé elvárnám az IBM-től egy nagyon jó fordító megírását.

IDE (integrált fejlesztőkörnyezet)

A gcc maga nem rendelkezik integrált fejlesztési környezettel, de létezik hozzá ilyen kiegészítés is. A Borland C++ a Windows környezetben megszokott IDE-t tartalmazza egy az egyben, amely — el kell ismernem — nagyon kényelmes. A Watcom C++ környezete már elmarad a Borlandétól, de azért még kellemesen használható. Sajnos a szerkesztőprogram hiányzik a csomagból, a Watcom erre a célra az EPM-et használja. Ez igencsak jól konfigurálható, de egy integrált szerkesztő megnövelné a termék értékét. A VisualAge-ben az IDE-nek a WorkFrame felel meg. Van hibakeresője, profilere, jó szövegszerkesztője és Visual Buildere, de ezek nincsenek igazán integrálva, nem úgy, mint a Borland C++-ban.

Debugger

Tulajdonképpen az összes debugger megfelelően működik, amit talán ki kell emelni az az, hogy a Watcom környezetben menürendszerrel működő karakteres módú debugger is található a grafikus mellett. A gcc-hez egy gdb nevű kissé „fapados" debugger tartozik, nincsen grafikus felülete, így az ehhez szokott fejlesztőknek nehézkesnek tűnhet. Ennek ellenére nagyon jó a hibakereső.

Összefoglaló

Nehéz dönteni a programok között. Mindegyik rendszer különböző feladatokra alkalmas. A Watcom nagyon jó fordítóval rendelkezik, a fejlesztőkörnyezete is könnyen használható, ráadásul a csomagban nem csak az OS/2, hanem a DOS és Windows alatti fejlesztőkészlet is megtalálható.

A VisualAge kiemelkedő része a Visual Builder, ahol a grafikusan megtervezett alkalmazás kódját generálja a program. Ez nagyban kiemeli a konkurens fejlesztőrendszerek közül. Sajnos hiányzik a csomagból az integráltság, és a fordító sebességén továbbá a generált kód méretén is lenne még javítanivaló. A csomag további előnye, hogy létezik már platformon is, azaz a megtervezett alkalmazás kódját Windows (vagy akár UNIX operációs rendszerekre is) alatt elkészíttetve nagyon egyszerűen portolhatók a programok.

A Borland C++ fejlesztését sajnos úgy tűnik abbahagyta a cég, a honlapon csak pár évvel ezelőtti kis technikai segítség található a programhoz. Kár, mert az IDE nagyon jól használható, bár a fordítót újra kéne hozzá írni és a fejlesztőkörnyezetben meglevő hibákat kiküszöbölni. A környezet tartalmazza az OWL-t (Object Windows Library), így az azt használó Windows programokat könnyű OS/2 alá portolni. Ezenkívül mem merem semmi másra ajánlani.

A végére maradt a gcc, de ez egyáltalán nem azt jelenti, hogy nem lenne jól használható program. A fordító jó, több kereskedelmi programnál is gyorsabb kódot generál (a tesztekből terjedelmi okokból kihagyott prímszámgenerátor esetén például a gcc lett az első), tartozik hozzá sok — a UNIX világban megszokott — segédeszköz. Tulajdonképpen minden célra megfelel, a legalkalmasabb azonban UNIX programok portolásához.

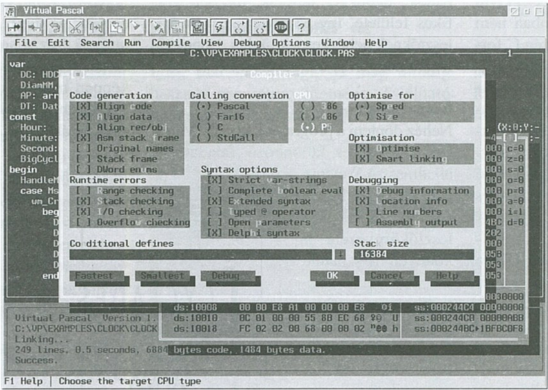
Tóth Ferenc

Virtual Pascal/2

Pascal nyelven programozók sem maradnak fejlesztőkörnyezet nélkül OS/2 alatt. Cikkünkben az egyik professzionális 32 bites Pascal fordítót mutatjuk be.

Egy nagyon jó fordító magában még nem elég a komoly piacok meghódítására — vélte a Virtual Pascalt fejlesztő fPrint UK cég, amikor úgy döntött, minél több, már piacon lévő termékkel teszi kompatibilissé szoftverét. Mindez nem bizonyult leküzdhetetlen feladatnak, ezt bizonyítja a termék legújabb, 1.10-es verziója, amely igen magas fokú Borland Pascal v7 és Delphi kompatibilitással rendelkezik.

A teljes csomag telepítve 27,4MB-ot foglal el a merevlemezen, ami manapság igazán nem mondható soknak. A példaprogramokat és segítségfájlokat a CD-n hagyva (onnan is használhatók, de ez a későbbi frissítés miatt nem ajánlott) csak 3,4MB(!) szabad hely szükséges. A termékkel 365 oldalnyi nyomtatott dokumentáció és néhány, közel 9MB helyet foglaló INF fájl jár, utóbbiak tetemes része az OS/2 API (Application Programming Interface, alkalmazásprogramozói felület) leírásával foglalkozik. A több mint 150 példaprogram szinte minden területet lefed: kezdve az elmaradhatatlan „Hello World" programtól az EA (Extended Attributes, kiterjesztett tulajdonságok) kezelésén és a multimédia programozásán át a gyermekfolyamat kimenetének átirányításáig.



1. ábra: A kódot akár Pentium processzorra is optimalizálhatjuk

A VP/2 csomag fontos része a magas fokú felhasználóorientáltsággal jellemezhető, környezetérzékeny segítséggel rendelkező integrált fejlesztői környezet, röviden: IDE (Integrated Development Environment), amely karakteralapú ablakozós technikát használ, és nagyon hasonlít a BP7-re. A Borland Pascal felhasználói a menüket, gyorsbillentyűket stb. az ott megszokottaknak megfelelően használhatják. Az integrált rendszer lényege, hogy a módosítás, fordítás, szerkesztés (linking), futtatás és hibafelderítés (debugging) valamennyi lépése egyazon felületen végezhető. A fejlesztőkörnyezet futtatható teljes képernyőn (ez PM nélkül is működik), OS/2 ablakban, PM ablakban, valamint a fordítónak van parancssoros változata is. A PM ablak többletszolgáltatása az ablak felső részén található eszköztár-ikonsor, amelyekkel a leggyakrabban használt funkciókat aktivizálhatjuk. A rendszer használatában járatos fejlesztőnek ez nem igazán jelent előnyt, mivel gyorsbillentyűk segítségével hamarabb célhoz érhetnek, mint egerészéssel, a program pedig az ablak teljes részében ugyanúgy karakteres felületen dolgozik.

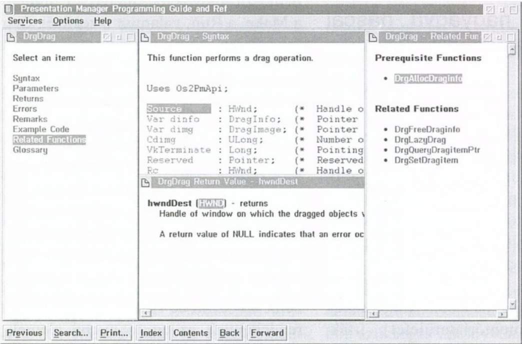
A VP/2-vel háromféle alkalmazást készíthetünk: csak teljes képernyőn futtatható OS/2 programot, OS/2 ablakban is futót és PM alkalmazást. Ezen programok az OS/2 minden erőforrását elérhetik, a szükséges API-k a megfelelő UNIT-ok használatával hívhatók meg. A meglévő Borland Pascal programok kis módosítással VP/2 alatt is fordíthatók, a fő különbség az, hogy itt a Word és az Integer típusok 32 bitesek (a 16 bites típusok neve SmallWord és SmallInt). A 32 bites „flat" memóriamodell számos korlátozást old fel, mint például a 64kB-os adatszegmens, a tömbelemek 64 ezerre korlátozott száma.

Említést érdemel a beépített linker (linkage editor), amely tipikusan 3-10-szer gyorsabb az OS/2 LINK386-nál, és annál kb. 10%-kal tömörebb kódot készít. Ha az erőforrásleíró-fájl (RC) neve a megegyezik Pascal forráskódfájl nevével, akkor amennyiben szükséges, a linker az erőforrás-fordítót is önműködően meghívja. További előny, hogy nem feltétlenül szükséges külön moduldefiníciós (.DEF) fájl készítése, hanem a Pascal forráskódban elhelyezett direktívák után megadhatók a szükséges adatok, pl. OS/2 ablakban futó program számára a STUB programot megadhatjuk így. (A STUB a keletkező .EXE fájl elejéhez csatlakozó DOS végrehajtható program, amely elindul akkor, ha a programot DOS környezetben próbálták futtatni. Rendszerint az feladata, hogy figyelmeztető üzenetet küldjön, ha OS/2 programot DOS alól indították.) Ez akár overlay-t alkalmazó DOS program is lehet, így akár olyan EXE fájlt is lehet készíteni, amely DOS és OS/2 alatt is működik. Ha nem írunk elő STUB programot, akkor a VP/2 olyan kódot állít elő, hogy az OS/2 program DOS ablakából is indítható lesz.

Hátravan még a minden, magára valamit is adó fejlesztői környezetből elmaradhatatlan hibakereső, amely természetesen az IDE része. Töréspontokat helyezhetünk el a forráskódban, de nyomon követhetünk vele már lefordított EXE fájlt is. Az intelligens, szimbolikus nyomkövető 14 különböző ablakkal rendelkezik. Ezek: Watch, Stack, Breakpoint, Log, Symbols, Units, Threads, Object Hierarchy, CPU, Dump, Registers, FPU és Exceptions. Újdonsága, hogy képes használni az OS/2 telepítő CD-jén található .SYM fájlokat. Ezek segítségével az OS/2 modulok nyomkövetésekor a változók és a belépési pontok nem memóriacímmel, hanem szimbolikus névvel láthatóak a hibakereső CPU ablakában.

A Virtual Pascal honlapja: http://www.fprint.co.uk/vpascal

VP/2-n kívüli egyéb szoftvercsomagokat is felhasználhatunk fejlesztéseinkhez: a Turbo Vision v2.0x (TV) és az Object Windows Library (OWL) a BP7-ből, a MATHPAK 87 könyvtár egy részét a Precision Plus Software-től és a shareware Techno Jock's Turbo Toolkitet a TJ Software-től. Ezen termékeket akkor is használhatjuk, ha nem vásároltuk meg azokat; ekkor forráskód szinten nem állnak rendelkezésre, de a VP/2-vel kapott, már lefordított DLL és OBJ fájlokat hozzászerkeszthetjük saját programunkhoz. A Turbo Power Software termékeit csak akkor használhatjuk, ha megvásároltuk őket (a VP/2 dobozában található kuponnal ezt 20% kedvezménnyel tehetjük meg). Ezek a karakteres felhasználói felület kialakítását segítő Object Professional, az adatbázis-kezelést támogató B-Tree Filer és a soros vonali kommunikáció programozását megkönnyítő ASync Professional.



2. ábra: Valamennyi OS/2 API leírása megtalalható az .INF fájlokban.

Mielőtt bárki is a VP/2 megvásárlása mellett döntene, érdemes kipróbálnia a CD mellékleten található teljes funkcionalitású demót, valamint elolvasnia a cikk szintén ott található kiegészítését. A már meglévő VP/2 tulajdonosoknak pedig számos kiegészítést, frissítést tartogat a korong. Ugyancsak nekik lehet hasznos a VP/2 levelezési lista, amelyre a majordomo@as60053.pc.nus.sg címre küldött *SUBSCRIBE VPASCAL* tartalmú levéllel tudnak feliratkozni.

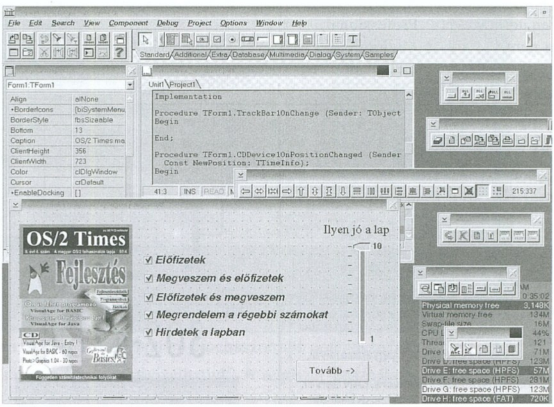
Varga László

Az OS/2 Times CD melléklete nem lemezmagazin. A rajta található cikkek nem azért kerülnek oda, mert értékük kisebb az újságban levőknél, egyszerűen csak nem fértek el a nyomtatott részben. Számos sorozat (hardverrovat, HTML tanfolyam, Photo> Graphics tanfolyam, OS/2 kérdések és válaszok gyűjtemény) terjedelmi okokból nem kerülhet az újságba. Nem mellékes az sem, hogy azzal, hogy ezek a cikkek a CD-re kerülnek és nem oldalszámbővítés révén az újságba, az olvasóink igen komoly többletköltségtől menekülnek meg.

SpeedSoft Sibyl 2.0

A Virtual Pascal mellett ez a másik nagyágyú pascal fejlesztőrendszer. Cikkünkben ezt vettük nagyító alá.

A SpeedSoft cég kiadta a Speed Pascal fordítóprogramok után a Sibyl névre hallgató vizuális fejlesztőrendszerét. A rendszer már a telepítéskor kísértetiesen emlékeztet Borlandék Delphijére, ugyanis a telepítés alatt végig antik görög szoborfejben gyönyörködhetünk. A rendszer használatához egyébként legalább 486-os PC, 800x600-as felbontást tűrő grafikus kártya és 8MB memória szükséges, ami manapság nem nevezhető túlzott igénynek (a szerzők a rendszer futtatásához azonban inkább Pentiumot ajánlanak). A 60 napos próbaverzió, amely a dokumentáción és a segítségen kívül szinte minden szolgáltatást tartalmaz, mindössze 7,5 (!) megabájtot foglal el a merevlemezen. (Ezt csinálják utánuk Borlandék!)



1. ábra: Az első értelmes(?) alkalmazás, ami az eszünkbe juthat

Az alapfelület menü- és eszközsávból áll. Az egész csupán a képernyő felső hatodát foglalja el. Innen végezhetők el a különféle műveletek, amelyekhez a megszokott felépítésű menüszerkezetben juthatunk el. A File menüben a szokásos műveleteket találjuk, innen nyithatók meg új forrásfájlok és itt menthető el a munka eredménye is. Az Edit menüben a megszokott műveletek (másolás, kivágás, beillesztés, visszavonás) mellett makrókat is rögzíthetünk, amelyek azután egy listába kerülnek, ahonnan tetszés szerint újra előhívhatók. A Search menüben a hagyományos keresések mellett növekményes keresésre és könyvjelzők beállítására is lehetőség van. A View menüvel jeleníthetők meg a szerkesztőrendszer egyes ablakai, ezekről a későbbiekben szólok. A Components almenüben lehet új összetevőket (Unitokat) felvenni a rendszerkönyvtárba, újjáépíteni a könyvtárakat és a párbeszédablakokhoz új mintákat készíteni. A Debug menü szolgál a hibakeresés vezérlésére. A szokványos hibakeresési funkciók mellett (lépésenkénti, illetve eljárásonkénti végrehajtás, változók figyelése, megállási pontok beállítása) megadhatók vizsgálódási pontok (Watchpoint) is, amelyek a megállási pontoktól annyiban különböznek, hogy itt megadható, hogy a program futása akkor álljon meg, ha a beírt kifejezésnek megfelelő memóriaterületet írták, olvasták, esetleg végrehajtották. Itt nyílik lehetőség a használt memóriaterület és a processzor megfigyelésére is egy-egy külön ablakban. A következő almenü a Project, ahonnan egy teljes programtervezetet kezelhetünk a létrehozástól egészen a végső fordításig. Az Options menüben igény szerint testreszabható a rendszer és megadhatók egyéb nyelvi beállítások is. Ez a menü azonban további pontokkal bővíthető a Tools ponttal, ahol bármilyen külső program megadható tetszőleges paraméterezéssel és névvel, például kedvenc formázóprogramunk megadásával az éppen aktuális forrásfájlt egyetlen paranccsal áttördelhetjük. A Window és a Help menüpontok teljesen szokványosak.

A menüsáv használatával a gyakori műveletek gyorsan és kényelmesen érhetők el. Ilyenek az Open, Save, Settings, New Form, a kereső, másoló és beillesztő műveletek, illetve a Compile, a Make, a Run, valamint a segítség és a rendszeren belül megnyitott ablakok listája. Mellette található az ablakelemek palettája, ahonnan kényelmesen felépíthető az ablakokat használó alkalmazások felülete.

A rendszerben beállítható a szerkesztő viselkedése, beleértve a tabulátorok és a kijelölt blokkok tulajdonságait is; az automatikus mentés paraméterei, a visszavonások száma, sőt, a kurzor alakja is megadható. Tetszés szerint állítható a szintaxis szerinti színezés, és itt nemcsak színt, de külön betűkészletet is rendelhetünk az egyes nyelvi elemekhez. Ízlés szerint adható meg, hogy a kulcsszavak milyen (kis- vagy nagybetűvel) kerüljenek a programkódba, és a forrás tabulálása hogyan a legkellemesebb a szemnek. A panelszerkesztő rácspontjainak elhelyezkedése, a kijelölő befoglaló téglalap viselkedése is testre szabható, csakúgy, mint az objektumböngésző tulajdonságai és egyes általános jellemzők is.

Elérkeztünk a Speed Soft Sibyl rendszer programozási és nyelvi lehetőségeinek tárgyalásához. Vizuális fejlesztőrendszerről lévén szó, szinte egyetlen programsor beírása nélkül készíthetők vele tetszetős alkalmazások. Új programtervezet létrehozása után a Sibyl azonnal felkínál nekünk egy kitöltetlen alapfájlt és egy unitot a kötelező kulcsszavakkal. Megnyílik továbbá az Object Inspector, amellyel csupán vidd-és-rakd-le módszerrel pillanatok alatt elkészíthetjük első ablakunkat a vezérlőelem-paletta segítségével. A képen látható példapanel is így készült, körülbelül öt perc alatt. Az egyes elemeket az Alignment Palette (elrendező paletta) segítségével ízlésesen „elegyengethetjük”. Az elkészült panel kódja automatikusan bekerül a unit forráslistájába, illetve az egyes elemek tulajdonságai a lefordított .SCU (Sibyl panel) fájlba. A rendszer képes

Delphiben készített paneleket is beolvasni. Az egyes panelelemek viselkedését eseményvezérelt módon adhatjuk meg, csupán hozzá kell rendelni a megfelelő eseményhez az előre elkészített eljárást. A panel kódja automatikusan mozgatható forrásfájlok között, így a programozónak nem kell feleslegesen energiát pazarolnia a kód elhelyezésére.

A programozási munkát a szerkesztő egyszerű és világos, jól áttekinthető kivitelezése is megkönnyíti. A Borland-környezethez szokott kezeknek sem kell új billentyűkombinációkat elsajátítaniuk, sőt, a hibakeresés használata is ugyanolyan, mint a Borland-féle környezeté. Megegyeznek a kezelőbillentyűk is (F7 - utasításonkénti, F8 - eljárásonkénti lépkedés). Itt meglepő, de hasznos lehet, hogy szükség esetén még a lefordított gépi kódú utasításokon is végig lehet lépkedni a memóriában (Ez fordul elő például, ha a SYSTEM.PAS fájl megléte nélkül próbálkozunk az F7 billentyűvel egy Begin kulcsszón).

A Browser jól használható az egyes objektumok, unitok és globális kifejezések szerkezetének megtekintésére és nyomon követésére. A kívánt kifejezés nevén duplán kattintva az ablak jobb oldalán megjelenik az ide tartozó eljárások, függvények, változók, állandók listája, attól függően, hogy mit szeretnénk látni.

A fordítás számos jellemzője állítható be a Sibyl rendszerben. Fordíthatunk OS/2 alá (ez csak természetes), de lehetőség van Win32 kód generálására is! Az is megadható, mikor írjon ki a rendszer figyelmeztető üzeneteket, és hogy az alkalmazás grafikus vagy csak egyszerű szöveges felületet használ-e, illetve mekkora a használt memória mérete. Rengeteg hibakeresési beállítás is rendelkezésre áll, és megadhatók a fordításhoz szükséges alkönyvtárak is. A fordítást beépített Speed Pascal 2.0 fordító végzi el. Egy grafikus felületet használó, csupán egyetlen ablakot tartalmazó lefordított alkalmazás mérete bizony elérheti a 300-400 kilobájtot is!

A Sibyl és a Speed Pascal 2.0 támogatja az adatbázis-kezelők használatát, a hálózaton keresztüli DDE-t (dinamikus adatátvitelt), az egy alkalmazáson belüli és az alkalmazások közötti vidd-és-rakd-le objektummozgatást, továbbá olyan kiegészítő egységeket tartalmaz, amelyekkel változatossá tehetők az alkalmazásablakok, mint például az átmenetes színezés vagy a hétvonalkás számkijelző.

A rendszer BIN alkönyvtárában a futtatható fájlok között található erőforrás-fordító (Resource Compiler) is, valamint a parancssori fordítók kedvelőinek kedvéért a Speed Pascal 2.0 ilyen változata is megtalálható.

Nagy Attila

Az OS/2 Times minden második hónapban jelenik meg, folyamatosan CD melléklettel. A megjelenés általában a hónap harmadik hetének keddjére esik. Ez évi tervezett megjelenési dátumaink:

október 14.

* * *

december 9.

PM programozás

Az OS/2 Presentation Manager programozása C-ben - Ablakállapot és egy kis grafika

Eddig arról volt szó, hogyan lehet létrehozni egy egyszerű Presentation Manager ablakot. A példaprogram létrehoz egy ilyen ablakot, megjeleníti benne a zászlót, majd vár a menükön vagy gyorsbillentyűkön keresztül érkező parancsokra. Nem sokat csinál, de lassan megpróbálunk valami hasznosabbat és érdekesebbet faragni belőle.

Ablakállapot

Az ablak, amelyet létrehoztunk, nem rendelkezik belső állapottal, vagyis viselkedése nem függ a múltban bekövetkezett eseményektől. Nagyon kevés hasznos állapotmentes ablak van. Az eddig felépített példaprogram nem sok mindenre használható, hacsak a képernyő „díszítését" nem tekintjük annak. A szerkesztés menü parancsai sem sokat csinálnak, talán a Másolás parancs bemásolhatná az ablak tartalmát a vágólapra. Ideje tehát kissé megváltoztatni a példaprogramot: ezen cikk végére egyszerű rajzolóprogram válik belőle.

Az ablak többféleképpen is tárolhatja állapotát. A legegyszerűbb módszer, ha statikus változókat hozunk létre az ablak üzenetfeldolgozóján kívül, és ezeket használjuk. Az egész addig működik, míg nem hozunk létre több példányt ugyanabból az ablakból. Például, ha létrehozunk egy nyomógomb vezérlőelem-osztályt, majd ennek több példányát, amelyek mindegyike egyedi állapottal és jellemzőkkel bír, az üzenetkezelőn kívül létrehozott közös állapotinformáció használata nem túl célravezető.

Szerencsére van jobb megoldás. Az ablakosztály bejegyzésekor meg lehet adni a rendszernek, hogy az osztály minden egyedéhez tartson fenn saját tárterületet. Itt foglalunk helyet egy mutatónak, amely az ablak állapotát tároló területre hivatkozik. Az ügyfélablak bejegyzése most a következőképpen történik:

```
WinRegisterClass(Horgony,
    Osztálynév, Üzenetfeldolgozó,
    CS_SIZEREDRAW, sizeof(void*));
```

Az állapotadatok címének tárolására szolgáló tárterület lefoglalása nem foglalja le és nem inicializálja az állapotadatok területét. Ezt az ablak létrehozásakor elsőként vett WM_CREATE üzenet feldolgozása közben kell megtenni. Az állapotinformáció lefoglalása tetszés szerinti módon (malloc, new stb.) történhet, de helyét a WinQueryWindowPtr hívással kell eltárolni, a következő módon:

```
struct Ablakállapot *Állapot =
        <tetszés szerinti lefoglalás>;
WinSetWindowPtr(hwnd, QWL_USER,
                        Állapot);
```

Miután ez megtörtént, az állapotinformáció címét le kell kérni minden egyes üzenet feldolgozása előtt, így:

```
struct Ablakállapot *Állapot =
            (struct Ablakállapot *)
WinQueryWindowPtr(hwnd, QWL_USER);
```

Na jó, erről ennyit. A példaprogramban egy kicsit más megközelítés látható. Az egyetlen lényeges eltérés az állapotinformáció lefoglalásában van.

Egy kis grafika

Az előző példában is volt egy kis grafikai programozás: az ablak tartalmának törlése, majd a magyar zászló megrajzolása. Most ennél egy kicsit nehezebb és hasznosabb program van soron: olyan ablak, amelyekben vonalakat lehet húzogatni az egérrel. Ez bemutatja a fentebb tárgyalt ablakállapot használatát is.

Ahhoz, hogy a program rajzolni tudjon az ablakba, először engedélyt kell kérnie a rendszertől, azaz meg kell szereznie az ablakhoz tartozó „megjelenítési tér hivatkozását" (presentation space handle). A megjelenítési tér (presentation space, PS) általánosságban olyan terület, ahová rajzolni lehet, és az OS/2 Presentation Managere alatt (általában) semmi köze a fizikai eszközhöz, amelyen az ábra végül meg fog jelenni. A PS azonban kapcsolódhat a fizikai eszközt leíró eszközkörnyezethez (device context, DC). Létre lehet hozni olyan PS-t is, amely nem kapcsolódik DC-hez, és csak később létrehozni a kapcsolatot, de ez már messzire vezet, és most nem foglalkozunk vele. Két, a mi céljainknak megfelelő módszer van megjelenítési tér létrehozására. Hogy melyiket használjuk, az a feladat függvénye. Ha programunk a rendszer Paint kérésére reagál, a használandó módszer a következőképpen fest:

```
RECTL Téglalap;
HPS hPS=WinBeginPaint(hwnd, 0,
                    &Téglalap);
// Ide jönne a grafika
WinEndPaint(hPS);
```

Amikor az OS/2 úgy dönt, hogy az ablak egy részét újra kell rajzolni (például az a felszínre kerül, láthatóvá válik, vagy mérete megváltozik), WM_PAINT üzenetet küld az ablak üzenetfeldolgozójának. A programnak meg kell hívnia a WinBeginPaint függvényt, hogy megkapja a megjelenítési tér hivatkozását. Ha az újrarajzolandó terület adataira is szükség van, a függvény harmadik paramétereként meg kell adni egy RECTL struktúra címét. Bármely normál grafikus beállító- vagy rajzolófüggvény használható a visszakapott hivatkozással (erről egy kicsit később lesz szó). A rajzolás végén a programnak meg kell hívnia a WinEndPaint függvényt, hogy felszabadítsa a megjelenítési teret.

A WM_PAINT kezelésétől eltekintve csaknem minden esetben az alábbi módszert kell használni:

```
HPS hPS = WinGetPS(hwnd);
// Ide jönne a grafika
WinReleasePS(hPS);
```

Ha megvan a megjelenítési tér hivatkozása, érdemes lehet megváltoztatni egyes alapértelmezett beállításait. Például, az alapértelmezett színkezelési módszer az IBM PC-k régi tizenhat színű szöveges módjához hasonlít, azaz a 0 jelöli a feketét, az 1 a kéket és így tovább, egészen a fehéret jelentő 15-ös számig. Én jobb szeretek valódi színekkel dolgozni, amelyeket a PC hárombájtos egységekben kezel: egy bájt a piros, egy a zöld és egy harmadik a kék összetevő számára. Minden egyes bájt 0-tól (teljesen sötét) 255-ig (teljes erővel világít) terjedő összetevőtartományt jelent. Ehhez a következő függvényt kell meghívni:

```
GpiCreateLogColorTable(hPS,
    LCOL_RESET, LCOLF_RGB, 0, 0, 0);
```

Ezzel a függvénnyel egész érdekes dolgokat lehet művelni a megjelenítési térben használt színkezeléssel, valamint az itt kinullázott paramétereket is lehet hasznos dolgokra használni, de példaprogramunk most inkább békén hagyja ezeket. Igazából ez a beállítás a legtöbb programhoz megfelelő, a bonyolultabb

esetekre csak akkor van szükség, ha a programnak sokszínű ábrákat színhelyesen kell megjelenítenie.

Amikor a program a WM_PAINT üzenetre válaszol, az első végrehajtandó művelet valószínűleg az újrarajzolandó téglalap háttérszínnel történő kitöltése lesz. Ezt végzi el a következő kódrészlet:

```
GpiSetColor(hPS, RGB_WHITE);
WinQueryWindowRect(hwnd, &Téglalap);
GpiMove(hPS,
          (PPOINTL)&Téglalap.xLeft);
GpiBox(hPS, DRO_FILL,
   (PPOINTL)&Téglalap.xRight, 0, 0);
```

Az első sor fehér színt állít be, az RGB színábrázolásnak megfelelően. A második sor megkérdezi a rendszertől a teljes ablak méretét. A harmadik a grafikus kurzort (hivatkozásként használt, láthatatlan pont) a téglalap bal alsó sarkába viszi, végül a negyedik kitölti a grafikus kurzor és a téglalap jobb felső sarka közötti területet. Lehet, hogy ez furcsának hat, de gyors és egyszerű módja az ablak adott színnel történő törlésének. A másik használható függvény, a GpiErase nem teszi lehetővé, hogy a programozó megadja a törléshez használt színt.

Azt mondtam, egyszerű rajzolóprogramot készítünk, amellyel egyenes szakaszokat lehet rajzolni az egérmutató vontatásával. Ehhez tudnunk kell, mikor nyomja le a felhasználó az egérgombot, mikor mozgatja az egeret és mikor engedi el a gombot, és ezt fel kell jegyeznünk az ablakállapotot tároló területen. Sőt mi több, az ablaknak meg kell jegyeznie a jelenleg kirajzolt vonalakat, hogy ha kell, újra tudja rajzolni magát. Ehhez egy kicsit elő kell készülnünk. Először is, definiálnunk kell az ablakállapotot tároló struktúrát (amely az eddig megrajzolt vonalakat tárolja). Kis gondolkodás után a következő struktúrát hoztam létre:

```
struct Ablakállapot {
  int GombLenyomva;
  int VonalakSzáma;
  POINTL Vonalak [ MAX_VONAL][ 2];
};
```

Az ablaknak tudnia kell, le van-e nyomva az egérgomb (mindjárt látni fogjuk, hogy miért), valamint ismernie kell a megrajzolt vonalak számát és végpontjait. Kezdetben GombLenyomva értéke hamis (FALSE), a vonalak száma pedig nulla. A következőt szeretnénk csinálni: az egérgomb lenyomásakor feljegyezzük az új szakasz kezdőpontját, a végpontot ugyanerre a pozícióra állítjuk. Az egérmutatót mozgatva a végpont koordinátáit folyamatosan frissítjük, amíg az egérgomb felengedését nem érzékeljük. Ekkor megjegyezzük ezt a végpontot, és kirajzoljuk a szakaszt. Ezt a három lépést kell ismételni (lenyomás, mozgatás, felengedés), amíg a vonaltár be nem telik (lásd MAX_VONAL, feljebb).

```
POINTL Pont;
Pont.x = SHORT1FROMMP(mp1);
Pont.y = SHORT2FROMMP(mp1);
Állapot->GombLenyomva = TRUE;
Állapot->Vonalak[ Állapot->
        VonalakSzáma][ 0] = Pont;
Állapot->Vonalak[ Állapot->
        VonalakSzáma][ 1] = Pont;
```

Az egérgomb lenyomásakor az üzenetfeldolgozóhoz a WM_BUTTON1DOWN üzenet érkezik. A program az üzenetből kiolvassa az egérmutató helyzetét, és ennek alapján frissíti ablakállapotát:

Az egérmutató koordinátáit az első üzenetparaméterben kapja meg az üzenetfeldolgozó, két 16 bites szám alakjában. Az ablakállapotot frissítjük, jelezve, hogy a gomb le van nyomva. Ha van még hely új szakasz számára, annak kezdő- és végpontját az egérkurzorra állítjuk. Az egér mozgatása közben az ablakeljárás WM_MOUSEMOVE üzeneteket fog kapni. A WM_BUTTON1 DOWN üzenethez hasonlóan ez is tartalmazni fogja az egérmutató helyzetét, amelyet a fenti példában bemutatott módon fogunk kihámozni belőle.

```
if (Állapot->GombLenyomva ==
                    FALSE) return(0);
hPS = WinGetPS(hwnd);
GpiCreateLogColorTable(hPS,
   LCOL_RESET, LCOLF_RGB, 0, 0, 0);
GpiSetColor(hPS, RGB_BLACK);
GpiSetMix(hPS, FM_INVERT);
GpiMove(hPS, &Állapot->Vonalak
        [ Állapot->VonalakSzáma][ 0] );
GpiLine(hPS, &Állapot->Vonalak
        [ Állapot->VonalakSzáma][ 1] );
if (Állapot->VonalakSzáma <
 MAX_VONAL) Állapot->Vonalak
   [ Állapot->VonalakSzáma][ 1] =Pont;
GpiMove(hPS, &Állapot->Vonalak
        [ Állapot->VonalakSzáma][ 0] );
GpiLine(hPS, &Állapot->Vonalak
        [ Állapot->VonalakSzáma][ 1] );
WinReleasePS(hPS);
```

Az első sor figyelmen kívül hagyja a felengedett egérgomb mellett történő egérmozgásokat. A következő utasításblokk lekéri a megjelenítési tér hivatkozását, a rajzolószínt feketére, a rajzolási módot invertálásra váltja (vagyis a kirajzolódó vonal az összes pont összes színösszetevőjét invertálni fogja, amellyel fedésbe kerül). Ennek két hasznos eredménye van: a kirajzolt vonal a háttérben lévő színtől függetlenül mindig teljes hosszában látható marad, illetve második kirajzolásra eltűnik. Az egérmutatót mozgatva olyan gumivonalat kapunk, amely a háttér felett simán mozog, de nem hagy nyomot maga után.

Amikor a felhasználó felengedi az egérgombot, az üzenetfeldolgozó WM_ BUTTON1UP üzenettel kénytelen szembesülni. Mint korábban, az üzenet most is tartalmazza az egérmutató helyzetét, amely ugyanúgy állapítható meg, mint azt korábban is bemutattam. Az üzenetre reagáló eljárás kódja:

```
if (Állapot->GombLenyomva ==
                    FALSE) return(0);
hPS = WinGetPS(hwnd);
GpiCreateLogColorTable(hPS,
   LCOL_RESET, LCOLF_RGB, 0, 0, 0);
GpiSetColor( hPS, RGB_BLACK);
GpiSetMix(hPS, FM_INVERT);
GpiMove(hPS, &Állapot->Vonalak
       [ Állapot->VonalakSzáma][ 0] );
GpiLine(hPS, &Állapot->Vonalak
       [ Állapot->VonalakSzáma][ 1] );
if (Állapot->VonalakSzáma <
  MAX_VONAL) Állapot->Vonalak
    [ Állapot->VonalakSzáma++][ 1] =
      Pont;
GpiSetMix(hPS, FM_DEFAULT);
GpiMove(hPS, &Állapot->Vonalak
  [ Állapot->VonalakSzáma-1][ 0] );
GpiLine ( hPS, &Állapot->Vonalak
  [ Állapot->VonalakSzáma-1][ 1] );
WinReleasePS(hPS);
Állapot->GombLenyomva = FALSE;
```

Mint korábban, az első sor most is a felengedett gomb melletti mozgatást szűri ki, míg a második a megjelenítési tér paramétereit állítja be. Az aktuális gumivonal ezután törlődik. A szakasz végpontja felveszi az üzenetben kapott értéket, és a vonalak száma eggyel nő. Végül, a rajzolási mód visszaáll az alapértelmezett egyszerű vonalhúzásra, és a szakasz végleges helyzetében kirajzolódik. Miután ez megvan, visszaadjuk a megjelenítési teret, és az ablakállapotban jelezzük, már nincs lenyomva a gomb. Ha a fentieket kiegészítjük a tárolt vonalakat újra kirajzoló WM_PAINT feldolgozóággal, jóformán kész is vagyunk a havi adaggal. Maradt azonban még egy probléma, méghozzá egy felettébb huncut példány: mi van akkor, ha az egérgombot lenyomjuk az ablak felett, majd a gomb lenyomva tartása mellett azt másik ablakra visszük át, és csak ott engedjük fel? Ha nem teszünk valamit, sosem értesülünk a gomb felengedéséről. Hogy ezt elkerüljük, meghívjuk a WinSetCapture rendszerfüggvényt, amint az egérgomb lenyomását érzékeljük, és ezzel magunkhoz ragadjuk az egeret, amíg a gomb felengedéséről nem értesülünk.

Rick Papo

REXX

programozási tanfolyam I. — Ismerkedés a REXX-szel

A REXX története

A REXX programozási nyelvet Mike Cowlishaw fejlesztette ki 1979-ben az IBM nagygépeken futó VM operációs rendszerre. A 80-as évek elején a REXX a VM rendszer részévé vált, s az IBM-től független cégek elkészítették a REXX Atari, UNIX, DEC, Amiga, Windows és DOS változatát is. Az OS/2 szerves részévé az OS/2 1.3 megjelenésével vált, s rövid időn belül nagy népszerűségre tett szert. Ennek az is oka volt, hogy a REXX-et a felhasználók véleménye alapján tervezték, s az elsődleges cél a felhasználó kényelmének biztosítása volt. Talán az egyetlen kellemetlen tulajdonsága a teljes neve: REstructured eXtended eXecutor (kb. átstrukturált és kiterjesztett végrehajtó).

A REXX előnyös tulajdonságai

A legtöbb REXX parancs közönséges angol szavakból áll, mint pl. a SAY vagy a PARSE, így könnyen tanulhatók és nem kell rövidítéseket vagy zsargonokat megjegyeznünk. Ugyanakkor a REXX tartalmazza a modern programozási nyelvekre jellemző strukturált felépítést, így a más nyelvek használatakor szerzett programozási tapasztalat könnyen átültethető. A kód formátumára nézve igen kevés megkötés van, s a felhasználó majdnem teljesen szabadon formázhatja programját. A REXX ún. interpretált nyelv, ami azt jelenti, hogy a megírt kód azonnal futtatható és nincs szükség a pl. C nyelv esetében elmaradhatatlan közbenső lépésre, a fordításra.

A REXX nagyon sok beépített függvénnyel rendelkezik, amelyekkel szinte minden feladatot (fájlok manipulálása, adatfeldolgozás, debuggolás stb.) megoldhatunk. Más nyelvek esetében sokszor magunknak kell megírni azokat a függvényeket, amelyek a REXX-ben eleve benne vannak.

Mivel a REXX már évek óta az OS/2 szerves része, ezért minden OS/2 vásárló automatikusan és ingyen hozzájut.

A REXX OS/2-es változata olyan kiegészítéseket is tartalmaz, amelyekkel pl. a WPS (Workplace Shell) objektumok és az OS/2 INI fájljai is manipulálhatóak. REXX programokkal könnyedén automatizálhatunk olyan folyamatokat, amelyek egyébként igen időigényesek. Aki pl. sokszor telepít rendszert és nem akarja a Munkaasztal egyedi beállításait minden esetben kézzel megcsinálni, annak érdemes a kívánt átalakításokat REXX programmá „fordítania”. Később aztán a programot futtatva bármikor pillanatokon belül elkészíthető a kívánt felépítésű Munkaasztal!

A REXX programok és a batch fájlok

Már az első alkalommal szembetűnik, hogy a REXX programok és a batch fájlok nagyon sok közös tulajdonsággal rendelkeznek. Azonos pl., hogy mindkét fajta állomány CMD kiterjesztéssel rendelkezik és hogy az OS/2 parancssorból futtathatók. A REXX azonban jóval több, mint közönséges batch-nyelv, s sokkal közelebb áll a jól ismert strukturált programozási nyelvekhez (BASIC, PASCAL). A közös kiterjesztés miatt azonban sajnos „kívülről” nem láthatjuk, hogy REXX vagy batch programmal van-e dolgunk. Ennek eldöntéséhez sajnos bele kell néznünk a kérdéses fájlba. Csak akkor van szó REXX programról, ha a fájl megjegyzés (comment) sorral (pl. /* Ez egy REXX program első sora */) kezdődik.

Felkészülés REXX programozásra

A REXX programozás legfontosabb kelléke a szövegszerkesztő. Elvileg bármilyen szövegszerkesztő használható, amennyiben ASCII módban mentjük el a kódot. Ennek ellenére az a legegyszerűbb, ha az OS/2 beépített szerkesztőit (e.exe, epm.exe) használjuk. A fejlett szövegszerkesztő (Enhanced Editor, epm.exe) különösen ajánlott, mivel REXX programozást támogató funkciókat is integráltak bele. A szövegszerkesztőn kívül természetesen szükség van még az OS/2 REXX komponensére is. Amennyiben az alapértelmezés szerinti beállításokkal telepítettük az OS/2-t, akkor a REXX támogatás adott. Ha nem, akkor a REXX komponenst a Módosító Telepítés (Selective Install) futtatásával tehetjük fel. Amennyiben nem tudjuk eldönteni, hogy jelen van-e a REXX támogatás, akkor írjuk be OS/2 parancssorba a VIEW REXX utasítást. Ha minden rendben van, akkor az online REXX segítség kell, hogy megjelenjen.

Az első REXX program

Indítsuk el szövegszerkesztőnket és gépeljük be az alábbi programot! Vigyázzunk arra, hogy a megjegyzés (első) sor a fájl legelső pozícióján kezdődjön, különben a parancsértelmező batch fájlként fogja programunkat kezelni! Az igazi kód tulajdonképpen a harmadik sorral kezdődik, ahol a program argumentuma (amennyiben létezik) a world változóba kerül. Az ötödik sorban kiíratjuk a Hello szót és a world változó tartalmát. Ha a programot argumentum nélkül hívtuk meg (tehát a world változó nem kapott értéket), akkor a REXX (jobb híján) a változó nevét fogja kinyomtatni. Az utolsó sor (exit) a programból való kilépést jelenti.

```
/* Az első REXX program */
If Arg() >< 0 Then ARG world
Say 'Hello 'world' !'
exit
```

Aki az EPM-et használja a szerkesztéshez, az megfigyelheti, hogy az If begépelése és a szóköz megnyomása után az EPM automatikusan beír még néhány szót. Ezt hívják az EPM automatikus kódkiegészítő funkciójának, ami első pillanatban inkább zavaró, mint segítő. A kódkiegészítés kikapcsolható, ha megnyomjuk a CTRL-I billentyűket és a megjelenő ablakocskában beírjuk az expand off parancsot.

Vegyük fel a kódot HELLO.CMD név alatt, majd pedig váltsunk OS/2 parancssorban ahhoz az alkönyvtárhoz, ahová elmentettük a programot és futtassuk a hello parancs begépelésével. Amennyiben helyesen gépeltünk be mindent, a képernyőn az alábbi választ fogjuk látni:

```
Hello WORLD!
```

Ha pl. az olvaso paraméterrel indítjuk, akkor a várakozásoknak megfelelő „Hello OLVASO!” választ kapjuk.

Gratulálunk első REXX programjához!

Kádár Zsolt

Ez a cikk egy 1 + 20 részesre tervezett REXX tanfolyam bevezető leckéje. A következő leckével indul az igazi tanfolyam, amikor is a REXX alapszabályairól lesz szó. Amennyiben arra megfelelő érdeklődés mutatkozik, az OpenBlue Bt. magyar nyelvű REXX kézikönyv kiadását is vállalja. Ha érdekli egy ilyen könyv, kérjük feltétlenül jelezze a Szerkesztőségnek!

OREXX

Sorozatunk hetedik részében az Object REXX többszálú képességeiről lesz szó

Multithreading

Az eddigi leírásokban ismétlődően találkozhattunk az OREXX azon tulajdonságaival, melyek lehetővé teszik azt, hogy olyan programokat készítsünk, melyek egyszerűen szólva „egyszerre több mindent csinálnak". Természetesen egy darab gépen egy darab processzorral ez elméletileg is lehetetlen (itt most ne említsük az OS/2 for SMP-t [Symmetric Multiprocessing] amely pont a többprocesszoros gépek ezen előnyét aknázza ki), de aki idáig eljutott, valószínűleg tapasztalhatta, hogy a látszólagosan egyszerre futó programok teljesen hétköznapiak lettek manapság, különösen az olyan operációs rendszerek esetén, mint amilyen az OS/2. Aki mégsem tudná, annak röviden pár szót a multitaskingról és a multithreadingről. A multitasking az, amikor több különböző program fut (látszólag) egyszerre, és ezek vagy kommunikálnak egymással vagy nem. Igazából a dolog úgy működik, hogy az első programból lefut egy „pici", majd a másodikból, majd az összes többiből, majd ismét az első jön ott, ahol abbahagyta, és így tovább. (Erre példa lehet egy szövegszerkesztő és egy zenelejátszó program egymás mellett futása.) A multithreading (a „thread" szó szálat jelent) ehhez nagyon hasonló módszer arra, hogy **egy** adott program egyszerre képes legyen többféle dolgot végrehajtani, mint például egy szövegszerkesztő szöveget formázni, nyomtatni és eközben kezelni azt is ha a felhasználó a program menüjében akar mászkálni. Az egymás mellett futó „szálak" pici programok a nagy programon belül, melyek többnyire közös adatokon dolgoznak. Természetesen itt is vannak buktatók, melyekkel szintén érdemes megismerkedni. Az egyszerre futó feladatok szinkronizálása nem egyszerű feladat, hiszen ezek többnyire aszinkron módon (vagyis egymástól nagyrészt függetlenül) futnak, és nem jósolható meg hogy amikor az egyiknek szüksége van a másikra az épp mit fog csinálni. Minden rendszer — így az OS/2 is — rendelkezik lehetőségekkel a szinkronizálásra, ilyenek a queue-k (sorok és vermek), semaphore-ok (magyarul szemafor, ami nem sokkal magyarabb, véleményem szerint) és néhány egyéb lehetőség. A lényeg, vagy inkább a fő veszély neve *deadlock*, amit én „végzetes patthelyzetként" fordítanék, és aminek a lényege az, amikor valami nem kellően átgondolt ok miatt az X program vár az Y eredményére, míg az Y eredményéhez szükséges az X kimenete. Így mindegyik a másikra vár, nagyjából a világ vége plusz mérési hiba ideig... Ilyen összeakadásokat bizony néha nagyon nehéz kivédeni, hiszen nem mindig egyértelmű a dolog: lehet hogy egy „belső mechanizmus" (mint a későbbiekben említendő guarding) akad össze egy programrészünkkel. Nem kell elszontyolodni ha kiderül egy ilyen hiba: magában az OS/2-ben is található néhány ilyen probléma, és valószínűleg minden többfeladatos rendszer rendelkezik ilyenekkel. Az OREXX minderre képes, méghozzá — valószínűleg ezt is említettem némi belülről fűtött szenvedély miatt is — igencsak egyszerűen. Példákat láthattunk eddig is, de nézzük kissé közelebbről a lehetőségeinket!

A legegyszerűbb — és így a legkönnyebben alkalmazható — módszer neve *early reply*, amely „korai válaszadást" jelent. Emlékezzünk a `return` utasításra, ami egy objektum visszaadott értékét adja meg, és többnyire jelzi az objektum programjának végét, hiszen az érték visszaadásával a programrész befejezte feladatait. Vagy mégsem? Az *early reply* lényege pont ez: az objektum ugyan visszaad egy értéket az őt hívónak, de ezután a futás két részre szakad: az egyik ágon a „szokásos" módon a végrehajtás visszakerül a hívóhoz az érték visszaadásával, és a program fut tovább. Azonban a másik ágon az objektumunk renegát módon folytatja futását a `return` után, és még számos érdekes dolgot vihet véghez befejeződéséig. Egyet nem tehet meg: többé nem adhat vissza értéket, legalábbis hagyományos módon nem. Ezen kívül azonban szinte bármit megtehet: hangokat adhat ki, szövegeket vagy grafikát szülhet a képernyőre, számolgathat a program változóival és hasonló nyalánkságok. Ezt egyszerűen úgy érhetjük el, hogy az elágazásba beteszünk egy `reply` utasítást, pont úgy, mintha oda egy `return` került volna, ami az objektum által visszaadott értéket adná meg.

```
/* threadek */
ah = .foo~new /* egy peldany */
bee= .foo~new /* es egy masik */
say ah~hush("Bim!")
/* fusson az elso */
say bee~hush("Bam!")
/* a masodik IS */
do 5 /* ket obj + foprogram */
  say "hmm..."
end /* ha vege, minden megall */
::class foo
::method hush
use arg blah
reply "Started" blah /* itt ketfele
agazik */
do 5
  say blah
end
```

A program igen egyszerű, mégis azonnal három szálon futnak az események (jobb mint egy kalandfilm! :-)), hisz fut a két objektum és a főprogram. Egyszerű, nem?

Kissé más megközelítéssel dolgozik a `start` metódus. Ez az Object Class metódusa, tehát mindenki rendelkezhet vele. Működése kissé összetettebb, de ettől egy egyszerűbb felhasználás esetén el is tekinthetünk. Mindenesetre amikor egy objektum a `start` üzenetet megkapja, akkor létrehoz egy névtelen Message objektumot, melynek paramétereként átadja saját magát, mint célállomást, valamint a startnak megadott paramétereket, mint az üzenet paramétereit. A Message objektumokat már ismerjük, velük lehet „aszinkron módon" (ami a többszálúság és párhuzamos végrehajtás alapja!) üzeneteket küldeni objektumoknak. Itt tehát az történik, hogy a Message objektum létrejön és „elindul útjára" mialatt a programunk fut tovább, függetlenül attól hogy az üzenet mikor él célba, vagy épp hogy az objektum akinek szólt az üzenet mikor óhajt befejeződni. De nézzük inkább az előző példát „startosítva":

```
/* threadek starttal */
ah = .foo~new /* egy peldany */
bee= .foo~new /* es egy masik */
ah~start("hush","Bim!") /* az elso */
bee~start("hush","Bam!")
                          /*a masodik */
do 5 /* ket obj + foprogram */
  say "...hmm.."
end /* ha vege, minden megall */
::class foo
::method hush
use arg blah
say "Started" blah
do 5
  say blah
end
```

Az eredmény nagyon hasonló, az eltérés

itt abban van hogy itt maga az indítás aszinkron (tehát független szálon futó), míg a reply-s példában pontosan látható a pont ahol elágazik az objektum futása. Látható, ez utóbbi módszer (a startos) közelebb áll a „hagyományos" multitasking gondolkodásmódhoz. Tehát ilyen egyszerű lenne a dolog? Fusson a program százfelé, semmi gond se lesz vele? Azért a dolog némileg több gondolkodást igényel hosszabb távon. Vegyük például az alábbi példát:

```
/* guard 1 */
ah = .brave~new
ah~start(cheeky) /* indul az egyik */
ah~start(spooky) /* es a masik */
do 6
/* es hogy lassuk hogy megy tovabb */
  say "hmm..."
end
::class brave
::method  init  /*  obj.  valtozo
kezdoerteke */
expose foo
foo = 1
::method cheeky /* szamolgat */
expose foo
do 4
  foo=foo+1
  say "Foo="foo
end
::method spooky /* csak szol egyet */
expose foo
foo=foo+668
say "Foo is" foo "!!!"
```

Amikor lefuttatjuk a fenti példát, valami furcsa történik! Miért nem indul el spooky? Látható, hogy szépen megvárja, míg cheeky befejezi a számolgatást, és csak utána indul el. De miért?

Az OREXX igyekszik minket megóvni a legtipikusabb hibáktól, melyek többszálú programok esetén előfordulhatnak, és ez a változók nemkívánatos egymástól függése. Vizsgáljuk csak meg, mi történne, ha spooky elindulna. Tegyük fel először is hogy a program nem csak zagyvaságokat írogat, hanem komoly számítási feladatokat végez változóin. Tegyük fel, hogy a program egyik szála éppen cheeky-t hajtja végre. Megnöveli foo értékét eggyel. Ekkor elindul spooky (mert odafönn nem szeret minket valaki) és megnöveli a foo-t 668-al! Cheeky mit sem sejtve továbblép, feltételezve hogy a foo eggyel nagyobb... ehelyett az valami egészen mást tartalmaz amit a program ott elvár, és ettől a ponttól megjósolhatatlan hogy mik történhetnek...

Ettől óv meg minket az OREXX úgynevezett *guarding* (őrzés, őrködés) módszere. Lényege, hogy egy thread futása blokkolt (vagyis nem indulhat el) addig, míg az általa használt nem lokális (helyi) változók közül egy is olyan, amit más thread használ, vagyis ne legyen olyan változó amin egyszerre többen is dolgoznak egymástól függetlenül. Ez a mechanizmus sok érdekes dologra használható, hiszen helyettünk ügyel arra, hogy az azonos változókat használó részek várják meg egymást, így egy jól megtervezett programban leveheti vállunkról a szinkronizálás nehéz problémáját (és mellesleg kizárja a változók ilyen használatából eredő *deadlock*-okat). Azonban sok esetben a programozók zokon veszik ha a gép túlságosan okos akar lenni. „Héé, gép, bízd rám a dolgot, én majd gondoskodom róla hogy ne akadjon össze semmi!" — mondhatja a programozó, és igaza van; végül is ő a főnök. Így tehát — mint az várható volt — a guarding kikapcsolható. De az OREXX fejlesztői látszólag nem a szegényes fantáziájukról híresek, ugyanis ezt ismét egy nagyszerű csavarral oldották meg.

Először is az alapértelmezés szerint az őrzőszolgálat aktív, gondatlan programozók önkínzásának megelőzése céljából. Ha valaki azonban teljesen ura a helyzetnek, az **unguarded** módosítót megadva egy metódus definíciójában az a metódus nem fog többet törődni azzal, hogy ki használja az általa is felhasznált objektumokat. (Ugyanezt dinamikus módon megtehetjük a Method class `setun guarded` metódusával, de ritka az mikor dinamikusan hozunk létre metódusokat, így ebbe most ne menjünk bele.) Azonban van egy érdekes lehetőségünk a védelem rugalmas szabályozására, és ez a `GUARD ON` utasítás. Számos formája létezik, ezekből pár példát nézzünk meg:

```
guard on /* innentôl él a védelem */
guard off
/* ezután pedig szabad a vásár */
guard on when z<>0
/* ha z nem nulla akkor legyen véde-
lem. értelme lehet pl. ha tudjuk,
hogy z=0 esetén nem fut le más objek-
tumban sem olyan kód, ami a mi válto-
zóinkat zavarhatja */
```

Látható, hogy nagyszerűen megállapíthatja magáról egy programrész, hogy épp olyan-e az objektumban a politikai helyzet, hogy futni kell, vagy jobb megvárni hogy mi történik. Ha a feltétel miatt a futás megáll, akkor a kifejezés minden esetben újra kiértékelésre kerül, amikor a benne szereplő változók értéke megváltozik. Amint a kifejezés hamissá (illetve `guard off when xxx` esetén igazzá) válik a futás azonnal folytatódik.

Nagyjából ezen a ponton értük el a tudásnak azt a mennyiségét, ahol látható a Message class értéke, hiszen ez nagyszerű eszköz az egymástól némileg függetleníthető részek indítgatásához, illetve ezek egymás közötti üzengetéséhez. Jelenlegi tudásunk alapján már nagyon sokféle feladatot meg tudunk oldani, legalábbis kétlem hogy a többszálú programok gyártásától bárkinek is kimelegedne az agya: tényleg nevetségesen egyszerű ezek létrehozása. A következő részben olvashatunk a „hagyományos" multitaskingot (több program fut egyszerre) segítő mechanizmusokról.

Gervai Péter

Az október 14-én megjelenő számunkban a következőkről olvashat:

Multimédia minden mennyiségben

MP3 lejátszóktól a rajzolóprogramokig minden, ami belefér. Embellish, TrueSpectra Photo>Graphics, CD lejátszók, hangok, videók, képek.

Játékok:

Links for OS/2, Stellar Frontier, Enterpreneur és még sok más! Quake kiegészítések (Capture the flag, gépjátékosok, fegyverek, kiegészítések, dokumentációk.)

StarOffice 4.0 első pillantásra

A CD-n

IBM hírek magyarul
Hardverrovat
Photo>Graphics tanfolyam II.
HTML tanfolyam II.
Humor
Újabb játékdemók
Multimédiás programok, shareware programok
Javítások, eszközvezérlők
...és még sok más újdonság - csak nálunk!

(A változtatás jogát fenntartjuk!)

A CD mellékleten megtalálja a StarOffice 4.0 béta 1 változatát!


VTCD VIDEOTON
Kompaktlemez-gyártó Kft.
Székesfehérvár, Aszalvölgyi u. 1.
8001 Székesfehérvár, Pf. 175
E-mail: vtcd@mail.datanet.hu
(06-22)329132
Fax:(06-22)329133
COMPACT disc DIGITAL AUDIO
COMPACT disc+ DIGITAL AUDIO
COMPACT disc DIGITAL VIDEO
COMPACT disc Interactive
COMPACT disc
Kompaktlemez
Kompakt Technológia
Kompakt Szolgáltatás